BUS CREW SCHEDULING USING MATHEMATICAL PROGRAMMING

by

Barbara Mary Smith

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

The University of Leeds
Department of Computer Studies
June 1986

Abstract

This thesis describes a bus crew scheduling system, IMPACS, which has been demonstrated to be successful for a wide variety of scheduling conditions, and is at present in regular use by three British bus companies, including the largest, London Buses Ltd.

The background to the bus crew scheduling problem is described, and the existing literature on methods for solution is reviewed.

In IMPACS, the crew scheduling problem is formulated as an integer linear programme, using a formulation which is an extension of set covering; a very large set of possible duties is generated, from which the duties forming the schedule are selected, in such a way as to minimise the total cost. The variables of the set covering problem correspond to the duties generated and the constraints to the pieces of work in the bus schedule.

For realistic schedules, it is impossible to generate all legal duties, and there are often too many pieces of work to allow each one to give rise to a constraint. IMPACS contains several heuristic methods which reduce the set covering problem to a manageable size, while still allowing good quality schedules to be compiled.

Techniques for speeding up the solution of the set covering problem have been investigated, and in particular a branching strategy which exploits features of the crew scheduling problem has been developed.

Acknowledgments

I should like to thank my supervisor, Mr. A. Wren, for his help and encouragement. I am also grateful to Mr. J.M. Elms, Schedules Manager of London Buses, and to Mr. D.N. Wilson, Mr. R.M. Wiseman and Mr. D. Smee of the Schedules Office, for their patience and co-operation during the IMPACS development programme.

CONTENTS

CHAPTER 1. INTRODUCTION

The bus and crew scheduling processes described here are concerned with the assignment of vehicles, drivers and conductors to work a bus company's regular daily operations. The scheduled service is fixed in advance, and will remain unchanged for usually months at a time; the buses operate over a relatively small geographical area, and at times during the day, the vehicles and crews return to the vicinity of the home garage, so that the crew can leave the bus, handing it over to another crew. The problems are therefore distinct from related problems such as delivery vehicle routeing, and vehicle and driver scheduling for long-distance coach operation.

Turner [1] in 1946 described the problems of bus and crew scheduling and general principles to be followed in compiling schedules. As far as is known, this is the only general description of manual methods of schedule compilation which has been published in this country, and it is still relevant to present-day conditions.

Since a bus can be run continuously from early in the morning until late at night, whereas bus crews can work for only about eight hours in a day and must usually be given a mealbreak off the bus during that time, in urban bus operations it is the usual practice to compile separate schedules for the buses and their crews. First, buses are allocated to the journeys to be operated, and then crews are allocated to the buses. Separate schedules are feasible because there is an opportunity to change the crew whenever the bus is at a suitable changeover point, and often this will happen at frequent intervals throughout the day.

1.1 Bus Scheduling

The bus scheduling process takes the set of predetermined journeys, to be made at specified times and between specified points, and allocates a vehicle to each trip in such a way that the total number of vehicles required to operate all the trips is minimised.

The potential efficiency of the bus schedule is clearly affected by the times and frequencies of the services to be operated, and the service planning process may to a greater or lesser extent be integrated with the bus scheduling process. For instance, the frequencies of journeys on fixed bus routes may be decided in advance, to give a planned level of service, but the precise times of journeys may be determined by bus scheduling considerations.

The journeys will in most cases run on the standard routes operated

by the bus company and usually, having arrived at the terminus of the route, the bus will wait at the terminus before making another journey back along the same route. Some of the journeys, on the other hand, may be individual contract journeys serving schools or factories, and since these journeys are usually geographically separate from the standard route network, the buses operating them have to run out of service or 'dead' to the starting point and from the finishing point.

When all the work has been allocated, dead runs are added to get each bus from the garage to the starting point of the first journey, and back again from the terminus of the last journey to the garage.

In some cases, the work will have been allocated to a single garage before bus scheduling begins, but in other cases, the work is shared by more than one garage, and the allocation of work between garages is part of the bus scheduling process.

The end result of the scheduling process is a bus schedule consisting of a set of running boards (or a number of sets if there is more than one garage) which indicate the journeys to be operated by each bus from the time it leaves the garage until it returns, together with any linking dead runs.

Normally, the bus is in continuous operation between leaving and returning to the garage, except for brief periods of idle time between journeys. Sometimes, however, it is convenient to park buses which are not required during the middle of the day at a site away from the garage. The two separate periods of operation count as one running board in bus scheduling terms, because the same bus is involved throughout, but in what follows they will be counted as two separate running boards, and references to taking a bus out of the garage or returning it to the garage should be taken to include parking a bus or picking up a bus from a bus-park.

1.2 Crew Scheduling

Whenever a bus is in operation, it must of course have a driver, and if necessary a conductor, assigned to it. Sometimes it is possible to leave a bus unattended for a few minutes, for instance at a bus station between the arrival of one journey and the departure of the next, but otherwise whenever a crew leaves the bus, a relieving crew must be immediately available to take it over.

Crew scheduling involves cutting up a set of running boards into a set of crew duties, in such a way that:

a) every part of every running board is assigned to a duty (except at times when the bus can be left unattended);

b) the duties conform to a set of agreed rules governing their legality and acceptability;

c) the number of duties required is minimised;

d) the total cost in wages is minimised.

The allocation of drivers, with or without conductors, to the bus schedule will throughout be referred to as crew scheduling, although two-person crews consisting of a driver and a conductor are becoming increasingly rare. A crew is always scheduled as a unit, and there is no essential difference between driver scheduling and crew scheduling, although in places where both types of operation occur, there are differences in payment and allowances for one-person and two-person operation.

Although every crew must be based at a home garage, there is no reason in theory why a crew should not work on a bus from another garage for part of their duty. The set of crew duties would then cover more than one set of running boards, one for each garage. This is common in rail crew scheduling, for instance, as described in section 2.7. In practice, however, bus crew schedules almost invariably cover the work from only one garage and this is true of all the schedules considered here.

For crew scheduling, it is necessary to know for each running board the set of times at which there is an opportunity to change the crew, so that one crew can leave the bus, to take a mealbreak or finish their duty, and another crew can take over the operation of the bus, having just started their duty or having already worked on another bus. Relief opportunities occur whenever the bus is at a convenient place for crews to change over; such places are called relief points. Hence, each potential relief opportunity has an associated relief time and relief point.

(The possibility that the bus may be left unattended creates a complication, because there are then two different relief times, one for the crew leaving the bus, and a later time for the crew taking over. However, a pair of relief times associated with a single relief opportunity will be referred to as one relief time throughout.)

The set of relief opportunities, with the associated times and points, is usually the only information required from the bus schedule in order to construct a crew schedule. It may sometimes be important to know as well which bus route is being worked at any time, for instance if there are

limitations on the time that a crew can work on certain routes, but in general the detailed information about the journeys operated by a bus is irrelevant for crew scheduling purposes.

The interval between one relief time and the next on a running board must be worked by a single crew, since by definition there is no opportunity to change crews except at relief times. Hence, this interval is, from the crew scheduling point of view, an indivisible piece of work. A running board consists of a string of these pieces of work, and although the start and finish of the running board are the same from both bus scheduling and crew scheduling perspectives, the relief times are in general quite different from the start and finish times of the journeys which constitute the running board from the bus scheduling point of view.

1.3 Interaction Between Bus And Crew Scheduling

Clearly the bus and crew scheduling problems are not independent, and in theory the overall least-cost set of schedules would be obtained by compiling bus and crew schedules simultaneously. However, this would be an extremely difficult problem, and it is not attempted, either by manual schedulers or by any of the computer scheduling systems which have been implemented.

In weekday bus operations, where there are well-defined peaks in demand in the morning and evening rush-hours, the number of buses required is the number needed at the height of the peaks, so that one of the main aims of bus scheduling is to schedule the peak periods as efficiently as possible. Similarly, an extra bus on the road during one of the peaks will require an extra crew to cover it. Hence, the most efficient peak usage of buses will also tend to allow efficient scheduling of crews, and to a considerable extent an efficient bus schedule is compatible with an efficient crew schedule, so that considering bus and crew scheduling as separate problems is not likely to increase the total operating cost by very much.

It is possible to take account of some of the obvious ways in which the construction of the bus schedule can adversely influence the subsequent compilation of the crew schedule. For instance, if a crew can only work for a maximum of four and a half hours before taking a mealbreak, then a bus which is out of the garage for five hours during the morning or evening peak will have to be covered by two crews, which would be very inefficient. Manual schedulers, and computer scheduling systems such as the VAMPIRES system [2] developed at the University of Leeds, try to avoid creating running boards which are slightly longer than

the maximum continuous driving time when constructing the bus schedule.

1.4 Construction Of Duties

A crew duty consists of one or more spells of work, each of which starts and finishes at a relief time. In the U.K., duties normally work on at least two buses, with an intervening mealbreak taken away from the bus.

Whenever a crew leaves a bus, unless the bus is returning to the garage at the end of the running board, or can be left unattended at that time, another crew must be ready to take over. Hence, a set of duties has to consist of spells of work which perfectly dovetail together so that all the running boards are completely covered, and at the same time, the spells have to fit together to form duties which conform to the agreed rules for valid duties. This combination of constraints makes crew scheduling a very difficult problem.

In many bus companies in North America and Europe, duties can work on just one bus without a break, for up to seven or eight hours. Usually the bus schedule is constructed so that such a duty contains several idle or layover periods of perhaps ten or fifteen minutes each, when the bus is waiting at a terminus, and this accumulated layover is accepted as a substitute for a mealbreak. Being able to construct a proportion of the duties in this manner greatly simplifies the task of constructing a crew schedule.

There are very few places in this country where a significant proportion of the duties work straight through on one bus. The only cases known to the author are Merseyside, where a mealbreak is notionally taken at the end of the duty, and Portsmouth. In other cases, a few duties may work on only one bus; for instance, some duties may have only one spell assigned to a specified bus with the other half left to be assigned by the garage inspector, or there may be some single spells to be worked as overtime. However, if only a few duties are involved, this does not have much effect on the difficulty of the problem.

1.5 Characterisation Of Crew Scheduling Agreements

The rules which valid duties must obey are partly determined by government and E.E.C. regulations, although the local agreement between the bus company and the union is usually more restrictive than the national rules. Since the introduction of new schedules is subject to negotiation, the formal union agreement may also have an accretion of unwritten rules, which schedulers must bear in mind in order to compile

acceptable schedules.

Union agreements vary considerably, but most contain rules similar to those listed below:

1. The crew start work by signing on at the garage. There is a paid allowance for signing on, plus an allowance for travelling from the garage to the relief point where the first bus is to be picked up, or for taking a bus out of the garage. There is a similar signing off allowance at the end of the duty.

2. There is a maximum length of time which the crew can work without a mealbreak: this will usually be about four to five hours. A stretch of duty, either from signing on to the start of the mealbreak, or from the end of the mealbreak to signing off, may be a single spell on one bus, or may consist of two spells with an intervening short break (a joinup). In the latter case, the joinup counts as working time. The length of the stretch always includes the time on the bus (and the joinup, if any), and may include the signing on or off allowance, and possibly some allowance at the mealbreak, e.g. for travelling to or from the canteen, or for taking a bus into or out of the garage.

The maximum stretch length may vary for different types of duty, or may be restricted if certain bus routes are involved.

3. The mealbreak must be at least a minimum length, which is usually a fixed amount (say 30 or 40 minutes) plus allowances for travelling to the canteen, etc., which depend on the relief points involved.

All of the mealbreak may count as working time, or none of it, or only part of it, such as the travelling time or any excess over a certain length.

4. The total working time in a duty (however defined) is limited.

5. The total spreadover of a duty, i.e. the time from signing on to signing off, is limited. The maximum may vary for different types of duty.

Although most scheduling agreements have these features, most have additional clauses as well, which vary considerably between agreements. It is not unusual for two agreements to contain contrasting clauses: for example, in some places, crews will only work on one bus route, whereas in others each duty should work on as many routes as possible.

1.6 Overtime

Under some agreements, it may be necessary to assign all the work in the bus schedule to standard duties each representing a full day's work. In other cases, there may be some provision for overtime, so that some of the work, usually consisting of single spells of about two to four hours, can be set aside to be worked by a crew in addition to their normal duty. The proportion of work to be covered by overtime may be fairly strictly defined in the agreement, or it may depend on the requirements of the schedule.

1.7 Split Duties

In urban weekday bus operations, there are peaks in passenger demand just before and just after the normal working day, as passengers travel to and from work. This demand is met by corresponding peaks in the number of buses in operation. There may be more than twice as many buses operating during the peaks as there are during the middle of the day.

In order to provide crews for the extra peak buses, most scheduling agreements allow a proportion of Monday to Friday duties to have a much longer spreadover than standard duties, say up to 12 hours or more, compared with perhaps 9 hours for other duties. The long spreadover is compensated for by a long break in the middle of the day, when fewer buses and hence fewer crews are required, and the maximum driving time is usually the same as for other duties (say about 8 hours). By using split duties, both the morning and the evening peak can be worked by one duty, so that a typical split duty would sign on at about 0700 and sign off at about 1900, with a break of three or four hours starting after the morning peak.

Split duties are normally unpopular with crews, and so attract premium payments; in consequence they are expensive to operate. Usually, there is a restriction on the proportion of duties which can be splits.

Although it is difficult to compile crew schedules for Monday to Friday operations which have significant peaks without using split duties, some agreements do not allow any split duties at all. The morning and evening peaks must then be worked by different crews, unless one peak can be worked as overtime before or after a normal duty; this in effect forms a duty with a long spreadover, but without the long break found in a split duty. In either case, there tends to be an over-provision of crews between the morning and evening peaks, so that duties may be very

short, or may consist of a peak spell together with an 'as detailed' spell in the middle of the day, i.e. the crew is technically on duty, but is only required as cover for other absent crews.

Conversely, where split duties are allowed, they can be useful even where there are no peaks in the bus schedule. For instance, an example given by Turner [1] and shown in Figure 1.1 demonstrates that split duties may be necessary to provide for mealbreaks off the bus. The number of buses on the road is constant for most of the day, but in order to cover the bus schedule in 12 duties, two split duties (Duties 6 and 7) are required. Turner points out that if mealbreaks off the bus were not necessary, the coverage by 12 duties could be achieved easily, without using split duties, by dividing each running board into two continuous duties.

1.8 Types Of Duty

Duties can be divided into split duties and the rest, sometimes called straight duties. The rules governing the formation of split duties are different from those applying to straight duties.

Duties may be further classified into different duty types. However, these duty types are usually classifications to be made after a duty has been formed, rather than restrictions on the formation of individual duties. For instance, the next stage after compiling a crew schedule is to draw up a roster, so that over a period of weeks, every crew works every duty in the schedule, and the duties follow each other according to some agreed set of rules: for rostering purposes, duties are commonly classified as early or late, so that crews work a week of early duties followed by a week of late duties. (This classification may also apply to split duties.) The agreement will then contain rules specifying allowed ranges of signing on and off times for early and late duties.

Although there is sometimes some variation in the scheduling rules as applied to different types of straight duty, normally the scheduler can form duties signing on and off at any time required by the schedule: the rules simply indicate whether a duty is to be classed as an early or a late duty. The classification into duty types affects the schedule as a whole, as opposed to the formation of individual duties. For instance, there may be a requirement that half the duties should be earlies and half lates to allow for a crew roster consisting of alternating early and late weeks.

Apart from any classification into types of duty laid down by the agreement, the constraints of the scheduling process often lead to the duties formed falling into a few recognisable types. From this point of

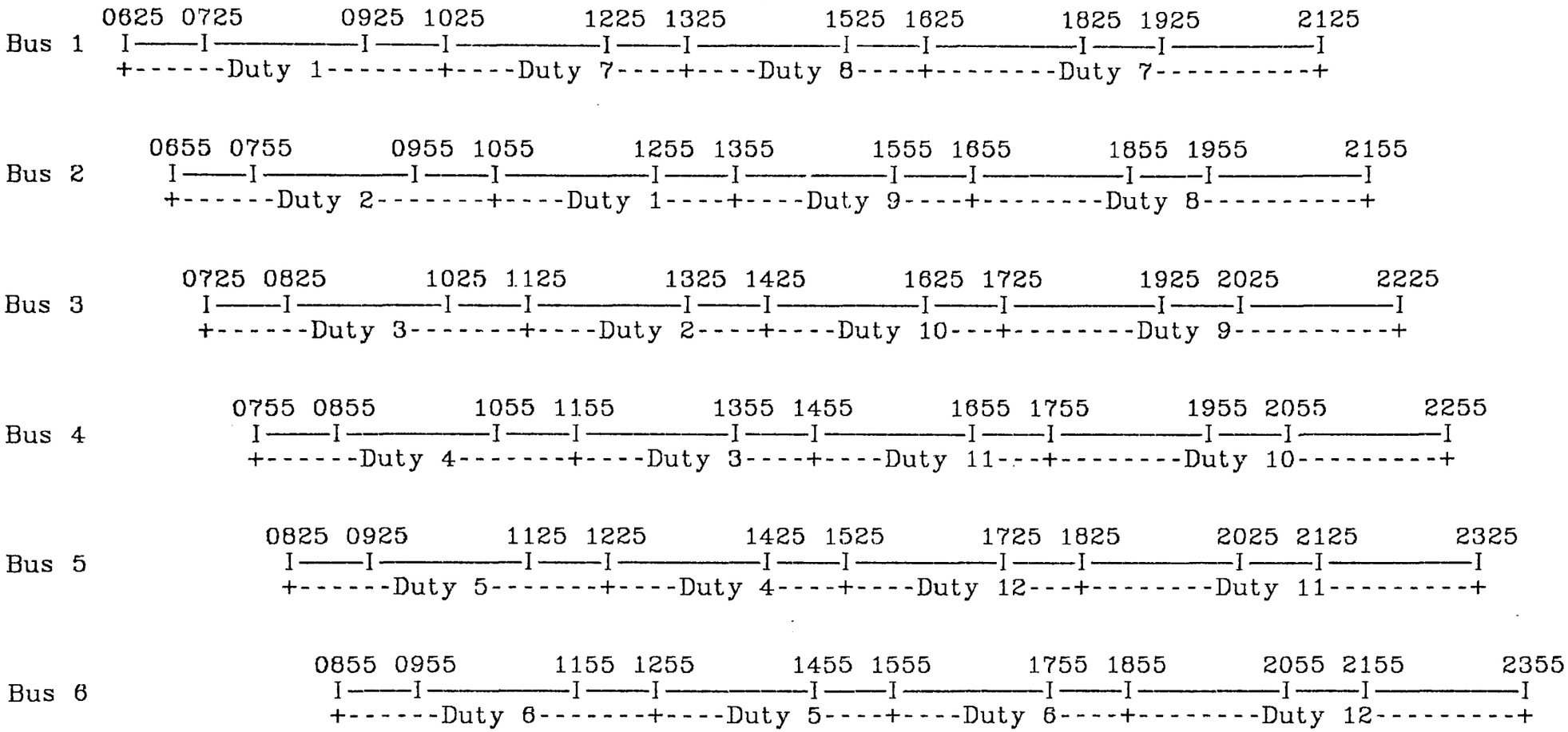


Figure 1.1 Example of a Crew Schedule

view, there are early duties, taking early buses out of the garage and finishing during the middle of the day, and late duties taking late buses into the garage late in the evening; usually these can also cover the evening peak. Many of these late duties form mealbreak chains, in which the crew returning from a mealbreak takes over a bus from a crew just about to start a mealbreak, ~~as shown in the diagram below~~. Eventually such a chain must come to an end, with a crew finishing a duty handing over to a late crew returning from a mealbreak. These duties finishing during the evening may be split duties, but more commonly are middle duties, which start during the middle of the day, often by taking over a bus from an early duty just finishing.

Figure 1.1 shows an example of a set of late duties (Duties 8 to 12) forming a mealbreak chain, terminating when Duty 12 takes over Bus 6 from the end of a split duty (Duty 6). In this artificial example, the early duties also form a mealbreak chain.

It may sometimes be necessary to take an early duty off the bus during the morning peak and start another duty then; these extra duties are day duties and will finish some time around the evening peak. There may also be other day or middle duties filling in gaps between early and late duties, particularly if there are no split duties.

Early, late and middle duties (and day duties, if required) will occur in both weekday and weekend schedules. Weekday schedules with morning and evening peaks will have split duties as well, if these are allowed.

1.9 Benefits Of Computer Scheduling

As already mentioned, compiling crew schedules is a very difficult task for manual schedulers because of the necessity to form valid duties which all fit together in an efficient way. In addition, with many agreements, there is a great deal of tedious arithmetic to be done in checking that the duties are valid and calculating the cost of the schedule. Constructing efficient schedules demands a high level of skill and experience, and is in any case very time-consuming. Schedulers are also responsible for constructing bus schedules and rosters.

Some large bus companies have centralised schedules departments, with well-trained and experienced staff, and the crew schedules produced are very efficient. Most bus companies, however, are too small to support a separate schedules office, and schedulers may often have other functions. In these circumstances, it is difficult for schedulers to develop the skills and experience required to compile efficient schedules, and they may not have sufficient time to devote to scheduling because of

other commitments.

Computer scheduling systems can be helpful for two reasons: they may be able to produce more efficient schedules than manual schedulers, and they may be able to produce results more quickly. These benefits apply to both bus and crew scheduling systems. The bus scheduling systems developed at the University of Leeds, VAMPIRES and TASC, described in [2], are becoming increasingly widely used in this country. VAMPIRES is a powerful program designed to produce very efficient schedules: savings of at least 5% can virtually be guaranteed in appropriate circumstances. TASC is designed to produce schedules of comparable quality to manual schedules very quickly.

On the other hand, there have so far been very few implementations of crew scheduling systems in this country. This is partly because the problem is difficult to solve by computer, as well as being difficult for manual schedulers. Even with a good computer method, it is very difficult to produce more efficient schedules than good manual schedulers, provided that they are given sufficient time. Considerable cost savings can sometimes be achieved, but they cannot be guaranteed, as they can be with the VAMPIRES bus scheduling system.

It is still true, however, that computer systems can produce good results more quickly than manual schedulers, although the benefits of this are hard to quantify. This factor is becoming more important, since bus companies are coming under increasing pressure for a variety of reasons (falling patronage, threatened competition, etc.) to revise services more frequently. Service revisions cannot be put into operation until the corresponding bus and crew schedules have been compiled, and in order to get the maximum benefit from the changes, it is important that the schedules should be as efficient as possible. The workload of schedulers is therefore increasing, and computer systems which can produce efficient schedules quickly should become more attractive.

Computer scheduling systems can be expensive to set up, and can consume large amounts of computer time, especially those which use mathematical programming methods. Whether this is an important consideration to a particular bus company depends on what computer facilities are available, whether there is spare capacity, how often new schedules are required, and many other factors. In cases where using a computer scheduling system can reduce the number of duties required to cover the work, the potential cost savings over the operating period of the schedule should easily outweigh the cost of producing the schedule. In other cases, where the main benefit of a computer scheduling system is

the ability to produce schedules more quickly, whether or not the benefits outweigh the costs depends very much on the individual circumstances of the bus company.

The IMPACS scheduling system which is described in chapters 3 onwards, has produced schedules for several different British bus companies which are considerably cheaper than the corresponding manual schedules. It is also being regularly used by the schedulers of London Buses Ltd. (formerly London Transport (Buses)): as described in chapter 9, the benefits in that case are mainly that the schedules are produced more quickly than manual schedules, which relieves the workload on the Schedules Office at a time when large-scale service revisions are being implemented at frequent intervals. It also allows a greater variety of options to be produced for each schedule, and the IMPACS schedules are often of better quality than manual schedules. Although the system uses mathematical programming methods and so is expensive to run, for London Transport the costs are worthwhile in view of these benefits.

CHAPTER 2. REVIEW OF COMPUTER SCHEDULING SYSTEMS

2.1 Introduction

Attempts to develop computer programs to compile bus crew schedules began about 25 years ago, with little success at first. By the 1970's, some progress had been made, and increasing interest in this field and the related field of bus scheduling was marked by an International Workshop held in Chicago in 1975. Two further workshops have been held, one in Leeds in 1980 and the third in Montreal in 1983, and the proceedings of these have been published ([3],[4]).

Several of the early investigations, such as Young and Wilkinson's work [5] in the 1960's, used various mathematical programming formulations of the crew scheduling problem, but the first successful systems used heuristic methods. In these systems, an initial schedule is constructed and then subjected to a number of refining techniques.

In recent years, the emphasis in heuristic systems has shifted away from the automatic production of a complete crew schedule and towards the use of a computer system as an aid to the scheduler. The justification given for this change is that schedulers prefer to be able to control the compilation process; it is also true that heuristic systems often do not produce good quality schedules without help from the scheduler. The latest generation of heuristic systems can all be used to build up schedules interactively and may depend largely on the scheduler's skill in order to produce a good schedule.

By the time of the Chicago Workshop in 1975, computers were sufficiently powerful for some progress to be made with mathematical programming methods, and interest now appears to be concentrated on these approaches rather than on heuristics. The most natural formulation of the crew scheduling problem seems to be a set covering or set partitioning model; a large number of possible duties are formed, from which a crew schedule is selected so as to cover all the work in the bus schedule at least once (set covering) or exactly once (set partitioning). Several computer systems which use these models or variants of them have been described; there are also a few which use rather different mathematical programming approaches.

Despite the increasing power of computers, the crew scheduling problem is much too large to solve exactly as a mathematical programming problem, except in extremely simple cases. All the mathematical programming systems employ some method of reducing the size of the problem to manageable proportions, and the main

differences between the various systems lie in the way in which this is done.

The number of successful implementations of computer scheduling systems is still small, especially outside North America. Many of the systems described have never been used except on an experimental basis; very few are currently in use by schedulers, and very few have been used by more than one bus company.

2.2 Heuristic Systems

The distinction between heuristic and mathematical programming methods for crew scheduling is not always clear, since the problems are almost always too large to solve optimally if a mathematical programming formulation is used, and so the problem is often broken down into subproblems, some of which may be solved using heuristic methods. Conversely, some of the heuristic systems use mathematical programming methods to solve subproblems which arise during the construction or refinement of the schedule. As far as possible the systems described below have been classified as mathematical programming systems if the problem is formulated in mathematical programming terms, even if the solution method is heuristic rather than exact.

2.2.1 Elias' Scheduling Systems

Elias' work on crew scheduling [6] in the early 1960's produced an aid to manual schedulers, to deal with the situation, common in the USA, in which many duties are straight runs, i.e. single pieces of work of about 8 hours in length. The remaining work is formed into split runs, consisting of two pieces of work with an intervening break, and some may be left as "trippers", that is, single pieces of work to be covered as overtime.

Elias' scheduling aid had three phases: the first split up the running boards or "blocks" into straight runs with a residue of pieces of work less than seven hours long. For blocks longer than 8 hours, a straight run was first cut from the beginning of the block. If there were no relief time giving a straight run exactly 8 hours long, as would normally be the case, the program chose a shorter or a longer run, whichever was the cheaper. The rest of the block was left as a piece of work, or if it was longer than 8 hours, another straight run was cut from the end of the block.

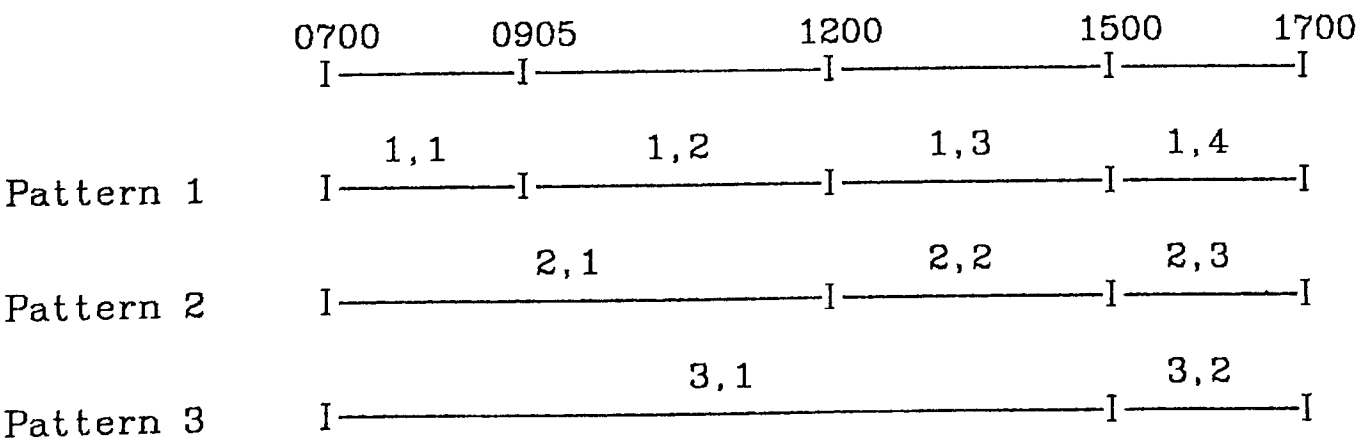
These steps were repeated, but cutting the straight runs first from the end of each block.

The scheduler was then presented with a list of alternative ways of breaking up each block, and chose a selection of straight runs. The remaining pieces of work were then combined in all possible ways by the second program to form a list of split runs. Again, the scheduler had to choose a set of split runs from this list. If there were too many single pieces of work left over, the scheduler could choose some of the straight runs from the first phase, and these would be split up and combined with the leftover pieces by a third program. This set of programs offered little help to the scheduler beyond calculating the costs of the candidate runs.

Later work by Elias [7] went much further and did produce complete schedules automatically. Part of the work developed a mathematical formulation of the problem, but it was too large to solve, and the method actually used was heuristic. The development of the mathematical model is described in [7] in terms of imagining a manual scheduler considering each block in isolation and considering potentially all the possible ways of splitting up the block into pieces of work. Then having decided how each block should be split up, the scheduler combines the resulting pieces into duties.

Although this might be a sensible way to work in the situation described above where most duties are single pieces of work, as described in [7] the method is clearly intended to be used when most or all runs work on two buses. In such cases it would not normally be sensible to consider splitting the blocks as a separate stage from combining the pieces into duties. The two-stage process described led Elias to an integer-programming formulation: two pieces of work linked together to form a duty are defined by 8 parameters ijkl,mnpq meaning that piece l of the k^{th} splitting pattern of block j on route i is linked to piece q of the p^{th} splitting pattern of block n on route m.

For instance, three possible splitting patterns for a block with four relief times are shown below; each piece is labelled by two indices, one for the splitting pattern, the other for the number of the piece in that pattern.



The integer variables of the formulation are:

$$x_{ijkl,mnpq} = \begin{cases} 1 \text{ if the duty is included in the schedule} \\ 0 \text{ otherwise} \end{cases}$$

There are four sets of constraints to ensure that all the bus schedule is covered and that for each block only one splitting pattern is chosen, and the objective is to minimise the cost of the duties selected to form the schedule.

This formulation gives an enormous number of variables and constraints even for tiny problems. For instance, Elias calculated that for a sample problem with only two blocks, the formulation would require over 100 million variables and nearly 70,000 constraints. Elias concluded, not surprisingly, that the problem was "well beyond the range of current computers", and turned to heuristic methods instead.

The heuristic approach which he adopted was based on a two-fold simplification of the mathematical formulation. First, the number of ways of splitting up the blocks was restricted, and secondly, having generated all possible duties from the pieces so formed, a simplified method was used to select a schedule. Only three ways of splitting up each block were allowed: in the example given in [7] the first piece of work on each block is approximately 3, 4 or 5 hours long, and the rest of the block is divided into pieces of about three hours each, the values being specified as parameters. Then all possible combinations of the pieces into two-piece runs are formed.

Many different schedules were constructed, by choosing a different starting duty for each one. The duties were ranked in order of cost and at each stage, the cheapest remaining duty which did not conflict with those already selected was added to the schedule, until no further duties could be added. Pieces of work not covered were formed into one-piece duties. The cheapest schedule constructed was offered to the scheduler.

Elias claimed that this method could produce schedules which were as good as those produced manually. Later work, described by Ward and Deibel [8], introduced refining heuristics which considered resplitting the blocks and interchanging pieces of work between two runs: only one initial schedule was produced because it was felt that the refinement would be able to produce a good result regardless of the starting point. Although none of the work was implemented, some of the ideas may have influenced the development of other heuristic systems.

2.2.2 TRACS

Work began on this system at the University of Leeds in 1967; it is described in several papers, the most up-to-date being Parker and Smith [9]. Although in outline the process is similar to that used by other heuristic systems, in that an initial solution is constructed and then improved by the application of various refining heuristics, the TRACS initial solution is more carefully constructed, since it was found through experience that the improvement techniques could not transform a poor initial schedule into a good one.

There are four stages in the construction of the initial solution:

a) Early duties are constructed. First, the latest time by which the first crew of each early bus must be relieved is marked. Duties are then formed in turn, at each stage the bus with the earliest remaining marked time being considered. A duty is formed on this bus, with a second piece taking over another bus at or before its marked time.

If the marked time on the first bus is not at the end of the bus, and the work following the marked time has not yet been allocated to a duty, it must be covered by an additional crew, since it is the earliest remaining marked time. It is better to have the additional crew starting as early as possible, so the marked time may be moved earlier.

A potential first piece of early duty can also usually be the first piece of a split duty: in the process of forming early duties, the program leaves enough first pieces unallocated to provide the desired number of split duties.

b) Late and middle duties are constructed in a similar way, working backwards from the end of the day. Evening peak work which is suitable for the second half of a split duty is not allocated.

c) First and second halves of split duties are paired up. Any remaining morning peak work is covered by extra early duties.

d) Unallocated work is attached to existing duties if possible.

The refining routines in TRACS are more powerful than those in other heuristic systems. One set of routines attempts to reduce the number of duties and unallocated pieces of work. Each duty is considered in turn to see if all the work in it can be accommodated in other duties. Unallocated pieces are reduced in length if possible, even if they cannot be eliminated completely, or are exchanged with shorter pieces of duty. Another routine redistributes work between duties so that duties with long spreadovers are given more work, in order to make short duties

lighter and so increase the chances of being able to accommodate pieces of duty or unallocated pieces of work.

The cost reduction routines are as follows:

a) each duty is cut in half at the mealbreak. Half of the duty is put into one list, and the other half into another. One list consists of first halves of earlies, splits and middles, with second halves of lates, and the other list contains the complementary halves. The pairing of the members of one list with the members of the other is solved as an assignment problem.

b) the times at which crews are changed are considered. A changeover is moved to an adjacent relief time if the cost is thereby reduced.

c) a small piece of work (not necessarily a whole piece of duty) is taken from each duty around the middle of the day, and the reallocation is solved as an assignment problem.

d) pairs of duties are considered to see whether the cost can be reduced by exchanging stretches of duty, or by moving a stretch of duty from one to the other.

The refining routines can be used in any order, and some may be used more than once.

Whereas the North American heuristic programs were initially developed to produce only one and two-bus duties, and three-bus duties were added at a later stage, if at all, TRACS formed three-bus duties from the outset.

The system was used to compile schedules for a number of different bus companies. The main obstacle to successful implementation was that it was found to be difficult to adapt the system to each new scheduling agreement. The system was designed so that it was easy to modify to meet the primary requirement, that the duties formed should be valid under the new rules. However, it was often found that in order to produce good schedules, it was necessary to make considerable modifications to the method of construction of the initial solution. Bus companies were usually reluctant to invest in the development work required to do this, as there could be no guarantee of ultimate success. When bus companies were prepared to make the investment, notably in the case of West Yorkshire P.T.E., good schedules were regularly produced.

Because of the difficulty of adapting an automatic crew scheduling system it was eventually decided that an interactive scheduling aid might be more acceptable to many schedulers. The COMPACS system described in section 2.2.6 was therefore developed in conjunction with Wootton

Jeffreys, a firm of transport consultants. The automatic features in COMPACS were based on the ideas used in TRACS, and TRACS itself has now fallen out of use.

2.2.3 RUCUS

RUCUS is a bus and crew scheduling system (the name stands for RUn CUtting and Scheduling) which was developed by the MITRE Corporation under the sponsorship of the U.S. Urban Mass Transportation Administration in the early 1970's. It was released for use by U.S. bus companies in 1975 and was then further developed by various consulting firms and individual bus companies, so that now several different versions exist. It was reported at the Leeds Workshop in 1980 that RUCUS had been installed at nearly 40 bus companies in the USA and Canada, but it has not proved entirely satisfactory, and UMTA sponsored development of a new version, RUCUS II, in the late 1970's.

The crew scheduling or run-cutting element of the original RUCUS system is a program called RUNS, which was described at the Chicago Workshop in 1975 [10]. The RUNS program uses heuristic methods, first constructing an initial schedule and then improving it. The stages in forming the initial solution are as follows:

a) create one-piece runs. These are duties in which the driver stays on the same bus throughout, without a mealbreak. They are a common feature of American crew schedules, although they sometimes occur only on the earliest and latest buses.

b) create "restricted-swing" two-piece runs. These are duties with an upper limit on the length of the mealbreak, i.e. two-bus straight duties. Early and late runs are formed alternately.

c) create "unrestricted-swing" two-piece runs. This step forms runs with mealbreaks of any length, which will mainly be split duties, although some extra straight duties may be formed. Similarly early and late duties alternate.

d) finally, any remaining unallocated work is added to the schedule as trippers, i.e. short one-piece duties.

The RUNS program requires the scheduler to set up the values of many parameters beforehand. Some of these are determined by the labour agreement, but others, such as the optimum length of a piece of work to be left behind when cutting a duty out of a running board, are not fixed and yet would have a significant effect on the quality of the initial solution. Wilhelm in [10] suggests that several program runs might be

necessary, with different parameter values, before an acceptable initial schedule is found.

Having reached this stage, a number of improvement algorithms are applied to the schedule. One set of routines attempts to eliminate the trippers, since these are undesirable and expensive. For instance, it may be possible to form two two-piece runs out of two trippers and one two-piece run. RUCUS does not handle three-piece runs, which restricts the potential of this routine.

The other type of improvement algorithm looks for reductions in the cost of the schedule. The costs considered can include penalties (positive or negative) indicating whether a duty is undesirable or desirable. This appears to be the only way of moving towards a good balance of duties. One of these algorithms considers each relief time which is the end of a piece of one duty and the start of a piece of another duty, and moves the changeover to the previous or subsequent relief time if the cost would thereby be reduced. The other algorithm considers duties in pairs, and looks at the effect of exchanging pieces of duty between them. These types of change are called shifts and switches respectively.

An optional extension to the RUNS program is available in cases where it is essential to eliminate all the trippers. However, the extension involves allowing illegal duties to be formed. The scheduler specifies which rules can be violated, and attaches penalties to each. For instance, Goeddel [11] describes the application of RUCUS to schedules of the Massachusetts Bay Transportation Authority. In this case, no trippers were allowed, and the final phase of RUCUS was required to eliminate them all. This resulted in several illegal duties, including a run with a spreadover of 14 hours 25 minutes, the maximum allowed being 13 hours. The paper does not mention how these illegalities could be got rid of.

Wilhelm claimed that the RUNS program gave the scheduler "maximum control over the schedule developed, and maximum flexibility in making changes to the schedule", via the user-specified parameters. However, the papers by Landis [12] and Hildyard and Wallis [13] at the 1980 Workshop contradicted this conclusion. Landis, describing the implementation of RUCUS in Portland, Oregon, said that RUCUS was producing good schedules there, but that considerable experimentation with the parameters had been required initially, in order to explore their effects on the final result and their interactions with each other. This experience is not portable from one bus company to another, since it is related to the labour agreement and the service pattern. Similarly, if the

bus company changes either of these, the quality of the schedules produced by RUCUS may decline unless a new series of experiments is carried out.

Hildyard and Wallis [13] described the work of SAGE, one of the main consulting firms involved in the modification and installation of RUCUS. Their main criticism of RUCUS was that it is hard to control, being run as a batch process. To run the program, the scheduler has to specify the values of about 50 parameters, originally with not very meaningful names. SAGE made RUCUS easier to control by providing a natural language for setting the parameter values and for running the program, and by splitting up the operation of the program into discrete steps, so that the scheduler can vary the sequence in which the schedule is formed and modify the parameters between stages. However, SAGE did not make any changes to the RUCUS algorithm.

Hence, although RUCUS allows the scheduler to modify the schedule produced by varying the parameter values, the mechanism for doing this is cumbersome, and the way in which the parameters affect the schedule may be hard to predict, so that without a great deal of experience, the scheduler is reduced to making random changes, rather than controlling the scheduling process, as claimed by Wilhelm.

2.2.4 RUCUS II

A preliminary version of RUCUS II was released in 1982 and was described by Luedtke [14]. The changes to RUCUS are mainly along the same lines as those made by SAGE. The parameters controlling the formation of the schedule are easier to specify, and can be modified as the schedule is being formed, and the scheduling process is interactive rather than operating in batch mode.

RUCUS II provides the scheduler with a number of commands to direct the formation of the schedule. Six of these form certain types of duty automatically, and roughly correspond to the steps by which RUCUS I forms the initial schedule. Two commands allow the scheduler to specify a one- or two-piece run to be added to the schedule, and two more delete an existing run. Presumably the deletion commands apply to both manually and automatically formed runs.

The remaining commands are used to improve the solution. One invokes the tripper elimination routine, as in RUCUS I, and another the cost reduction routine, which looks for possible switches and shifts. There are also manual commands to switch pieces of work between specified duties, or to shift a specified changeover earlier or later.

The heuristics employed by the automatic commands appear to be the same as in RUCUS I, except that there are two versions of each of the commands which form early and late two-piece straight runs. One pair of commands is more sophisticated than the other, in that they attempt to dovetail the duties formed, by constructing mealbreak chains.

The reason for having two commands operating in different ways appears to be to allow the scheduler to produce different solutions. "The complete set of runcutting steps can be applied in varying sequences, with or without adjustments to default parameter values, thereby permitting the scheduler to employ differing strategies for quickly generating several alternative runcut solutions and reporting the estimated costs associated with each." Luedtke describes a number of alternative schedules produced for one problem, using different combinations of the automatic commands, including the improvement commands. The balance of duties differed considerably between the different schedules: one had four straight runs, eleven splits and six trippers, whereas another had ten straight runs, four splits and seven trippers. However, of the five schedules produced, the one which was judged best was produced using some of the manual commands as well as the automatic ones, and had only three trippers, suggesting that the tripper elimination routine is not very successful.

Presumably, a scheduler could become skilful in applying the commands provided by RUCUS II, with practice, although experience in manual scheduling might not be very relevant. At present RUCUS II does not provide the scheduler with as much control over the schedule formation as does COMPACS: it appears that the duties formed by the automatic commands are added to the schedule immediately, and not presented to the scheduler for approval first, although they can be deleted one by one using manual commands.

Future developments planned for RUCUS will give the scheduler more control over the operation of the automatic commands, and at certain points allow the scheduler to override the choices being made.

At present three-piece runs are not formed by the automatic commands. Multi-piece runs can be formed by the manual commands but are then ignored by the optimization routines.

Ball, Bodin and Greenberg [15] report on enhancements to the improvement routines used by RUCUS and RUCUS II. There are two improvement routines, as described above: one considers all possible exchanges of pieces of work between duties and makes the switch if the schedule is thereby improved; the other considers all relief times at

which one duty hands over to another and shifts the changeover to an adjacent relief time if the schedule is improved. The IMPROVE command in RUCUS II first looks for beneficial switches, then for shifts, and repeats this until the solution cannot be improved further.

Ball et al. point out that this procedure does not guarantee to find a globally optimum set of switches or shifts. The new switching algorithm which they propose for RUCUS II does solve "the problem of finding a globally optimum set of one- and two-piece runs given a set of pieces" by using a matching algorithm. The algorithm will also handle side constraints on the ratio of straight runs to split runs and on the total number of trippers, using Lagrangean relaxation.

The proposed shifting procedure on the other hand does not find a globally optimum set of shifts, but simply considers all the changeovers on a block at once, rather than a single changeover. The problem of finding the best set of shifts for the entire block is formulated as a network flow problem. This is done for each block in turn, which, provided that improvements have been found, creates a new set of pieces. The optimum matching of the new set of pieces is then found, and so on until no more improvements can be made.

The new IMPROVE module was applied to two test problems, and indicated that, within RUCUS, it gave substantially better results than the current version, both in cost and in balance of duties.

2.2.5 SRI's IMPACS System

The system described by Howard and Moser [16] of SRI International, also called IMPACS, is an interactive crew-scheduling system which offers several levels of automation: at the simplest level, the scheduler assigns duties manually, using the computer only to check the legality of each duty, and store the current state of the schedule. At the opposite extreme, the schedule can be formed automatically, using heuristic methods.

As usual with heuristic methods, refinement techniques are included in the system so that the initial solution can be improved. One of these techniques looks for combinations of adjacent pieces of duty and ranks the possible combinations according to the likelihood of being able to match the resulting piece of work with another piece to form a two-piece duty. (The system does not at any stage consider forming three-piece duties.) The scheduler then chooses which pieces are to be combined. This technique appears to be the principal means of reducing the number of duties in the schedule.

The other two refinement techniques described occur commonly in heuristic systems. One considers each relief time at which crews change over, and uses an adjacent relief time instead if the schedule is thereby improved. Howard and Moser put forward an extension to this technique, in which all possible moves to adjacent relief times are considered simultaneously, using a set partitioning formulation. The constraints correspond to the end trips of each existing piece of duty, and the columns correspond to the duties which could be formed from the existing duties by moving some or all of the ends of the pieces to an adjacent relief time. They planned to solve this new problem using an L.P. relaxation, and suggested that in most cases the relaxed solution would be naturally integer. Judging by experience with set covering scheduling formulations, this is a reasonable supposition.

The second refinement technique keeps the pieces of duty fixed, and finds the optimal matching of these pieces into duties, again using a set partitioning formulation. This is an extension of a commonly-used refinement technique in which exchanges of pieces of duty are considered. This had been implemented, and Howard and Moser reported that solving the relaxed L.P. was very fast, and non-integer L.P. solutions were rare.

The SRI-IMPACS system had been under test by the New York City Transit Authority since 1981, but had not been implemented by 1983. The tests demonstrated that a 6% saving in payroll costs could be achieved, through an increase in the number of split duties and a decrease in the number of straight duties. However, the change in the balance of different duty types was not accepted by the drivers initially, and it is not clear whether the schedules were in fact put into operation. It is also unclear whether the system would have shown any savings compared with the manual schedule if it had been constrained to use the customary balance of split and straight duties, nor indeed whether the system could be constrained in this way.

2.2.6 COMPACS

The Wootton Jeffreys' COMPACS system was described in [17], although a few improvements have been made since then. It originated in an interactive crew scheduling program, TRICS, developed at the University of Leeds. COMPACS also incorporates a modified form of the heuristics used in the TRACS system (section 2.2.2) to build up an initial schedule.

The TRICS system at its simplest allows the scheduler to specify

duties one by one until the schedule is complete. The program checks that each duty is valid and keeps track of the work already covered, so that the scheduler does not inadvertently assign the same work to two duties. At a more sophisticated level, the program can suggest several alternative ways of completing a duty outlined by the scheduler, with a cost assigned to each, and the scheduler may select one of the alternatives.

The COMPACS system offers the scheduler a wide range of options, from building up the schedule one duty at a time, as in TRICS, to forming a complete schedule automatically as in TRACS. At any stage, the duties already formed can be unassigned or edited. The program can also be run in a semi-automatic mode, in which the scheduler asks for a certain number of duties to be formed, given the current unassigned work, and then inspects them before deciding to accept, reject or modify them.

Throughout, the system is simple and easy to use, with screen-based editing and menu selection of options.

Since COMPACS only constructs an initial schedule and does not have any refining routines, when operating in automatic mode the system is not as powerful as TRACS.

COMPACS has been tested by several of the users of the Wootton Jeffreys' bus scheduling system, TASC, since this can automatically create the relief time file required by COMPACS and hence reduce the effort of data entry considerably. Some schedulers are happy to use the interactive facilities to modify the duties suggested and to control the scheduling process themselves; others prefer a completely automatic system, in which case the schedules produced by COMPACS are often not as good as manual schedules.

2.2.7 The Ravenna System

Martello and Toth [18] describe a system which is in operation in Ravenna, Italy; it is described as a heuristic system, although it makes use of mathematical methods in part.

The bus schedules in Ravenna and other Italian towns have a lunchtime peak comparable with the morning and evening peaks. In fact, in the sample schedule given in [18], the number of buses on the road reaches its maximum at lunchtime. In addition, every driver whose duty covers the whole of the lunchtime period must have a mealbreak then; the driver must have a mealbreak of at least 1:15 falling entirely within the period 1100 to 1500. The allocation of drivers over lunchtime is further constrained in that at most 10% of drivers can have a lunchtime

mealbreak. Because it is so important to schedule drivers as efficiently as possible over the peaks, these rules will clearly have a considerable effect on the number of drivers required, and the quality of the final schedule will depend on how well the allocation of drivers over the lunchtime peak is done.

There is also a rule requiring that drivers working from before 1900 to after 2100 must have a mealbreak then, but as this is well after the end of the evening peak, and there is no restriction on the number of drivers taking evening mealbreaks, this rule does not have such a drastic effect on the efficient coverage of the bus schedule.

Martello and Toth point out that because no driver can work continuously over the whole of the lunchtime period, any bus which is on the road for the whole of the period requires two drivers to cover it, but that two buses can be covered by three drivers if there are suitable relief times allowing a driver to work on the first bus, take a mealbreak and then return to work on the second bus.

This is an example of a mealbreak chain, as discussed in section 1.8. In fact, with the values given in [18], it would be possible to fit in two consecutive mealbreaks over the lunchtime period, and so cover three buses with four drivers, but Martello and Toth are only concerned with looking for matchings of pairs of buses and not in looking for longer chains.

Running boards which are not suitable for matching may be made so by aggregating them with others, so that the work of two aggregated running boards can be covered consecutively by the same driver and the aggregated board is then suitable for matching. (Running boards considered for aggregation either start or finish during the lunchtime period.)

The maximum number of possible aggregations is found by forming a graph and solving a Bipartite Cardinality Matching Problem, and the maximum number of possible matches is found by solving a Cardinality Matching Problem. However, at lunchtime, the number of matches which can be made may be limited by the number of lunchtime mealbreaks allowed.

The scheduling method described in [18] uses the aggregation and matching procedures just described to allocate drivers to the mealbreak periods in an efficient way, and otherwise uses a greedy heuristic to construct a duty schedule by progressively adding work to existing duties or introducing new duties if necessary.

First an efficient coverage of the evening mealbreak is found, and the corresponding duties are introduced. The work after the evening mealbreak is assigned to the late duties, whose mealbreaks have already been determined. The afternoon work, between the lunchtime and evening mealbreaks, is first assigned, as far as possible, to the late duties. The possible matchings over the lunchtime period are found (but no duties are assigned), and unmatched running boards are assigned if possible to duties covering part of the evening mealbreak interval.

The duties with lunchtime mealbreaks are introduced, and any remaining afternoon work is assigned to these duties. As much as possible of the morning work, before lunchtime, is also assigned to them. Any remaining morning work is assigned to duties finishing during lunchtime, or if necessary to new duties.

Finally a refining procedure is applied, which improves the duties by "local exchanges", presumably exchanging pieces of work between duties. The system is described as interactive, but it is not clear to what extent schedulers are involved in the compilation process. It is reported to be in use in Ravenna and producing better results than the manual algorithms.

The heuristics used to cover the work outside the mealbreak periods do not appear to be at all sophisticated, and it is clear that the system only produces good results because it has been specially designed to deal efficiently with the tight mealbreak constraints and because in Ravenna the treatment of the mealbreaks is so important. If the mealbreak constraints were significantly relaxed, it is doubtful if good schedules would be produced, so that this is far from being a general-purpose scheduling system. On the other hand, a general-purpose scheduling system could probably cope quite well with the Ravenna conditions. Indeed, one of the sample schedules mentioned in [19], which were compiled using the HASTUS system described in section 2.4.3, was a Ravenna schedule. Any system using a set covering or a set partitioning formulation, as described in the following sections, is helped rather than hindered by tight constraints, since the number of possible duties to be considered is automatically limited, and the sizes quoted in [18] for two sample schedules (38 and 58 duties) would not be beyond the capabilities of a system such as IMPACS.

2.3 Set Covering And Set Partitioning Models

These formulations of the crew scheduling problem involve generating by some means a large set of possible duties, each with an associated cost and from which a subset of duties can be selected to form a crew schedule in such a way as to minimise the total cost.

The set covering problem can be expressed as the integer linear programming problem:

$$\text{minimise} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad i=1,2,\ldots,m$$

$$\text{and} \quad x_j = 0 \text{ or } 1 \qquad j=1,2,\ldots,n$$

As a formulation of the crew scheduling problem, the m constraints correspond to the pieces of work in the bus schedule, and the n variables correspond to the duties of the generated set, so that $a_{ij} = 1$ if the j^{th} duty covers the i^{th} piece of work and c_j is the cost associated with the j^{th} duty. The variable $x_j = 1$ if the j^{th} duty is in the schedule, 0 otherwise.

The constraints indicate that each piece of work must be covered by at least one duty. If the inequalities are replaced by equalities, it becomes a set partitioning problem, and each piece of work must then be covered by exactly one duty. These models are described more fully in chapter 5.

In principle, the set of duties forming the variables of the model could contain all possible legal duties; if it could be solved, the solution to the I.L.P. would then be the minimum cost schedule. However, unless the number of pieces of work in the bus schedule is extremely small, the total number of possible combinations forming legal duties is very large, and the resulting I.L.P. could not be solved. Hence, all the systems which use one of these models have some means of reducing the number of possible duties considered.

The number of constraints which can realistically be handled in an I.L.P. is a few hundred at most. Since each constraint corresponds to a piece of work, defined as the interval between two consecutive relief times in the bus schedule, one would expect that in urban bus operations, with fairly frequent relief times, there would be too many constraints in the I.L.P. unless some reduction was done here as well.

Few of the systems using a set covering or partitioning model mention this problem and explicitly reduce the number of pieces of work used in the formulation; one which does is the RATP system described in

section 2.3.1. In other cases, it is clear that the way in which the number of possible duties is reduced will at the same time implicitly reduce the number of pieces of work, because many of the relief times in the bus schedule cannot be used by any of the remaining duties.

An alternative way of dealing with the size problem is to decompose the schedule into smaller units: an example of this approach is described in section 2.3.4. Otherwise, even with some explicit or implicit reduction in the number of constraints, the systems described below can only compile relatively small schedules.

2.3.1 The RATP System (Paris)

The first successful crew scheduling system using a set covering or partitioning model was that developed for the Paris bus company, Regie Autonome des Transports Parisiens, which was described by Heurgon [20] at the Chicago Workshop in 1975. This uses a set partitioning formulation.

The formulation has two side constraints:

$$\sum_j x_j = \mathrm{N}$$

$$\sum_j t_j x_j \leq \mathrm{D}$$

where N is the required number of duties in the schedule, t_j is the length of the break in duty j, and D is the maximum total length of all the breaks, apparently derived from a maximum average spreadover defined for each schedule.

The RATP crew scheduling problems are relatively small, since each route is scheduled separately, with between 5 and 45 buses operating a route. A large proportion of the duties are continuous duties without a mealbreak, including the early and late duties. The minimum proportion of continuous duties in a schedule varies between 30% and 100%, depending on the type of schedule. The requirement for continuous duties makes the construction of crew schedules much simpler than the normal U.K. conditions.

Even so, the problem is too large to be solved if the initial set contains all possible duties. Heurgon describes three ways in which the problem size is reduced:

a) certain duties are determined in advance by the scheduler. These are apparently mostly continuous duties.

b) every relief time is given a code between 0 and 5, according to the likelihood that the relief time will be used in the schedule. The allocation

of codes appears to be done by the scheduler, who then selects the code numbers to be allowed in a particular run.

c) extra restrictions on the formation of duties are added to those in the union agreement, for instance, minimum duty lengths and maximum lengths of breaks.

The set partitioning problem is first solved as a continuous L.P. problem, and then if necessary an integer solution is found by implicit enumeration or a cutting plane method: in a difficult problem, both methods may be used. Heurgon reports that 30% of the continuous L.P. solutions are integral, and that if the constraint on the total length of the breaks could be dropped, the proportion would rise to two-thirds.

Implementation began in 1973, and was intended to be completed by 1978. However, there were delays, partly due to variations between routes, and partly because scheduling is not centralised in RATP: each route is the responsibility of a separate route manager, which makes implementation of a computer system difficult. The system was eventually abandoned.

2.3.2 DIGOS (Amsterdam)

Borret and Roes [21] describe a test of a system, originally designed to schedule train guards, on a bus route operated by the Amsterdam Urban Transport Company. The Saturday, Sunday and Monday to Friday schedules were compiled for this route.

Initially, nearly all possible duties were generated. It would normally be impracticable to do this, but the pieces of work in the test schedules were 2 hours long on average, which is unusually long for urban bus operations, and the late duties worked continuously on one bus without a mealbreak, so that they could in effect be specified in advance.

Even so, they found that the total number of possible duties for the Monday to Friday schedule was 30,000 (based on 200 trips). To reduce this number some restrictions were placed on the formation of three-bus duties, and most of the split duties were formed as two separate half-duties. To produce the final schedule, the half-duties in the solution were matched by hand. These restrictions reduced the number of possible duties from 30,000 to 2,000.

A set partitioning formulation is used, with a number of side constraints:

a) the number of early halves of split duties must equal the number of late halves of split duties;

b) the average working time must be less than a specified maximum;

c) there may be limits on the number of duties of a particular type.

A standard mathematical programming package, MPSX, was used to solve the problem. Borret and Roes reported good results, except for the Saturday schedule when the restrictions described above were applied to the duty generator.

This was clearly a very limited experiment and it seems likely that the system would run into serious problems if applied to routes with more frequent relief times, or to agreements without the simple structure for late duties applying in Amsterdam. It is believed that little further progress was made with the DIGOS system.

2.3.3 The CRU-SCHED System

The CRU-SCHED system was designed and implemented for the Dublin City Services of the Irish bus company Coras Iompair Eireann, using an extension of the set partitioning and set covering models. It is described by Mitra and Darby-Dowman ([22],[23]) and, in an earlier version, by Mitra and Welsh [24].

The formulation is governed by the particular requirements of CIE; the schedulers wish to specify the exact number of duties of each type in advance, and hence by implication the total number of duties in the schedule. There is then no guarantee that a feasible solution exists, unless some work can be left uncovered, and the formulation allows for this.

The generalised set partitioning model proposed by Mitra and Darby-Dowman is:

$$\text{minimise} \quad \sum_{j=1}^{n} c_j x_j \; + \; \sum_{i=1}^{m} w_i^u u_i \; + \; \sum_{i=1}^{m} w_i^o o_i$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \; + \; u_i \; - \; o_i \; = \; 1$$

$$x_j = 0 \text{ or } 1 \qquad j = 1,2,\ldots,n$$

$$u_i = 0 \text{ or } 1 \qquad i = 1,2,\ldots,m$$

$$o_i \geq 0 \text{ and integer} \qquad i = 1,2,\ldots,m$$

where the u_i and o_i are slack and surplus variables respectively, representing under-cover and over-cover, with associated costs w_i^u, w_i^o.

This formulation is further modified in CRU-SCHED, so that the only objective is to cover the bus schedule in the required number of duties:

$$\text{minimise} \quad \sum_{i=1}^{m} w_i^u u_i \; + \; \sum_{i=1}^{m} w_i^o o_i$$

with additional side constraints to ensure that there are the specified number of duties of each type, and that the average working time of the duties of each type is less than a specified maximum.

The number of possible duties generated is limited by imposing additional rules to eliminate duties which although valid are unlikely to appear in the schedule; a similar technique is used in IMPACS, as described in chapter 4. Mitra and Darby-Dowman report that this limitation is sufficient to ensure that the set of duties generated is of manageable size. The system does not deal with three-bus duties, which also limits the number of possibilities.

The authors state that "the system employs a special purpose integer programming code involving the use of heuristics and a special branching strategy that leads to acceptable computer run times." No further details of the solution method are given.

If the solution has pieces of work which are uncovered, the scheduler must take further action; the problem may be modified by amending either the bus schedule or the rules governing the validity of the duties, or both, and then reprocessed. Alternatively, the solution can be amended manually to incorporate the uncovered piece(s) into the existing duties. The amendment of the duty rules reflects another special feature of CIE scheduling: the rules are somewhat flexible and a degree of illegality is allowed so long as it is not too blatant.

The system is designed to be easy to use by schedulers; for instance, a screen-editing system is provided for data input and amendment.

The CIE schedulers were reported to be making good use of the system and producing schedules with between 10 and 50 duties, and this must be counted as one of the few successful implementations of a crew-scheduling system using mathematical programming. However, the features which have been built in to cater for the peculiarities of CIE might make it difficult to adapt the system for other situations. For example, most schedulers would not want to specify the exact number of duties of each type in advance, as required by CRU-SCHED, and the fact that the costs of individual duties do not appear in the objective would not usually be acceptable; it is not normally sufficient to achieve coverage of the bus schedule in the minimum number of duties, regardless of the quality of the duties used.

2.3.4 A Decomposition Approach

Ward, Durant and Hallman [25] suggest decomposing the crew scheduling problem by type of duty, so that the construction of late duties is a separate problem from the construction of early duties, and so on. Their reasoning is that the subproblems should then be sufficiently small to solve as set partitioning problems without further simplification.

[25] describes in detail the formation of late duties. The cost of each duty has an element related to its productivity, defined as the ratio of paid time to driving time. A dummy (slack) variable is introduced for any piece of work which would have to be covered by a middle duty if not covered by a late duty, and these are given a high cost to minimise the number of middle duties required.

The method was applied to sample schedules from three different bus companies, and a set of late and middle duties were compiled for each. The number of late and middle duties produced was the same as in the existing schedules, and Ward et al. claimed that the productivity of the duties compared favourably with the existing late and middle duties.

The potential difficulty of this approach is that the duties of the final subproblem may not be very good because of the restrictions imposed by the duties already formed. Unfortunately, the work reported in [25] had not progressed as far as producing a complete schedule, and it is believed that little further work was done, so it is not clear whether the approach would have been successful.

2.3.5 Ryan and Foster's Work

Ryan and Foster [26] describe a restriction of the variables in the set partitioning problem to ensure that the solution to the relaxed L.P. is naturally integer. This was based on their earlier work on the vehicle scheduling problem (Foster and Ryan [27]).

The vehicle scheduling problem is to design a set of routes from a central depot to deliver goods to customers at specified locations, subject to any restrictions on the capacity of the vehicles, duration of routes, and so on.

Foster and Ryan showed that the vehicle scheduling problem can be formulated as a set partitioning problem, with variables corresponding to feasible routes and constraints corresponding to customers. However, because the total number of possible routes is extremely large, they proposed forming only routes of a certain form, called "petal routes". The number of variables is thereby reduced to a manageable level, and

Foster & Ryan showed that the solution to the relaxed L.P. is nearly always integral.

A petal route operates over a sector of the region centred on the depot; all customers within the sector are served by the route. Routes of this form can be generated by ordering the customers radially around the depot and selecting contiguous subsets of the customers in this ordering, subject to the constraints on vehicle capacity and so on.

The effect of the petal route structure on the integrality of the relaxed L.P. arises from the fact that if there were a break in the radial ordering of the customers so that no route included both customer C_j and customer C_{j+1}, then by reordering the customers to start at customer C_{j+1}, every column of the matrix $A = (a_{ij})$ would have the form $(0,0,\ldots,0,1,1,\ldots.1,0,0,\ldots.0)'$. Foster & Ryan show that in that case the matrix A is unimodular, i.e. every square submatrix of A has a determinant of 0, 1 or -1, which is a sufficient condition for every basic feasible solution of the relaxed L.P. to be integer.

Foster & Ryan discussed possible ways of relaxing the restriction to petal routes in order to obtain better solutions to the vehicle scheduling problem, while still maintaining the near-integrality of the L.P. When fractional values occurred during the convergence of the L.P., cutting planes were introduced to restore integrality. They reported that the results obtained on a set of test problems compared favourably with other methods.

Their later paper on crew scheduling [26] describes an analogy to the petal structure of feasible routes. A one-bus duty is directly analogous to a petal route, the ordering of the pieces of work being the order in which they appear in the bus schedule, i.e. time order on each bus in turn. A minor modification of this structure to allow each piece of work to be followed by its unique first available subsequent piece of work, even if this involves a change of bus, still maintains the unimodularity of the A matrix. However, this is not a sufficient generalisation to allow feasible duties to be formed, in most cases, because there is no provision for a mealbreak.

Ryan & Foster claim that if a "modified first available" structure is used, in which a duty does the next available piece of work after the completion of the mealbreak, fractional solutions are still infrequent in the convergence of the L.P. Unfortunately, the schedules obtained are not very good, and they propose various relaxations to allow more duties to be formed, without losing the near-integrality of the L.P., if possible. For instance, there may be pieces of work which are not the "first available"

for any other piece of work, and hence these pieces of work will not appear in any duty except as the first piece of work. This is acceptable if the pieces occur at the beginning of the day, but otherwise, the first available structure should be relaxed to include second and third (or even higher) availables for some pieces of work, to allow these pieces to appear.

An alternative relaxation strategy is to examine the schedule formed using the pure first available structure and remove the work covered by the best duties. The remaining subproblem is solved again, still using the first available structure, which would involve using higher availables in terms of the original problem. This can be repeated until all the duties are fixed. They suggest that this is similar to the way in which some manual schedules appear to be constructed (based on schedules from the Auckland Regional Authority, New Zealand) and that the scheduler could make the selection of best duties from each solution, thus building up the schedule interactively.

For cases in which the L.P. solution is not naturally integer, Ryan & Foster use a branch-and-bound technique, using a special branching strategy; this was incorporated into the IMPACS system and is described in detail in chapter 6.

Although the ideas described were applied to some test schedules from Auckland, the system was not implemented, and not much more practical work was done. The interest of the work lies in the light it throws on the factors which influence the integrality of the relaxed L.P.

2.3.6 The A.T.A.C. System (Rome)

Piccione, Cherici, Bielli and La Bella [28] describe a crew scheduling system using a set partitioning formulation, which had been implemented for the Rome bus company, A.T.A.C.

It appears that although two-bus duties are allowed in Rome, most of the work is covered by one-bus duties and overtime pieces: a sample schedule in [28] has eleven duties, of average duration 5:30, of which only three work on two buses, and eight overtime pieces. The crew schedules for each route are compiled separately, which makes each problem relatively small, although usually still too large to be solved without some reduction in the number of variables.

Each bus is first considered separately and cut into duties (presumably one-bus duties) and overtime pieces. From these duties a minimum length and maximum cost are computed, and the values used to eliminate many of the total set of feasible duties formed by

considering the whole bus schedule at once. The duties formed on the individual buses are an initial solution to the problem, and Piccione et al. state that many of the duties in this solution can be fixed, so that again the total set of feasible duties can be reduced, by eliminating any which cover work assigned to a fixed duty.

By these means the number of variables in the set partitioning problem is reduced to between 1000 and 1500. An optimal solution is found using a modification of the Garfinkel and Nemhauser algorithm.

The method described appears to depend crucially on the existence of a large proportion of one-bus duties, so that although successfully implemented in Rome, it is not generally applicable, and in particular would not be suitable for U.K. conditions.

2.4 Other Mathematical Programing Systems

2.4.1 The Hamburg System

The crew scheduling system developed by the Hamburger Hochbahn Aktiengesellschaft (HHA) was described at the Chicago Workshop in 1975 [29]; a later version was described at the Leeds Workshop in 1980 [30]. Hoffstadt in [30] reported that since 1979 all HHA schedules had been produced by the system.

The HHA union agreement allows for continuous duties, i.e. duties working on one bus without a mealbreak, and these are preferred to duties which work on two or more buses. Hence, many of the duties can be cut as one piece from the start or end of a running board, without difficulty.

The combination of shorter pieces of work into two-bus duties is solved as a series of assignment problems: first for the early duties, then for the late duties, and finally for the split duties. In each case, two lists of pieces of work are formed, and the maximum number of combinations of elements from the two lists into valid duties is found.

Although successfully implemented in Hamburg, the system is of only limited interest in the U.K. context because of its dependence on the existence of a large proportion of continuous duties.

2.4.2 Manington's Work

Manington [31] investigated a linear programming approach to bus crew scheduling, using a method similar to the set covering formulation, in that a set of possible duties is constructed and a schedule selected from this set in order to cover all the work at minimum cost. However,

the formulation he used contains several additional sets of constraints, which reflect the way in which the original set of duties is constructed.

A subset of duties is constructed for each crew, where a crew is considered to be assigned to either the first piece of work on an early bus, or to the last piece of work on any bus, the subsets being mutually exclusive. There are n subsets, of which e are assigned to the first pieces of early buses.

Then in addition to the usual set covering constraints requiring that each piece of work should be covered at least once, the formulation contains the following constraints:

$$\sum_{j=1}^{l_i} d_j = 1 \qquad i=1,2,\ldots,e$$

$$\sum_{j=1}^{l_i} d_j \leq 1 \qquad i=e+1,\ldots,n$$

where $d_j = 1$ if the j^{th} duty for crew i is in the schedule, 0 otherwise, and l_i is the number of possible duties for crew i.

The first set of constraints ensures that the first piece of work on every early bus must be covered by exactly one of the crews assigned to that piece. The last piece of work on some buses may be covered by the end of an early duty, so that it may not always be necessary to choose one of the duties specifically assigned to that piece of work, and hence the second set of constraints are inequalities, and fewer than n duties may be required to cover the schedule.

The formulation had to be extended because not all duties cover the first or last piece of work on a bus. The extension proposed involves deciding exactly when such duties should start or finish: for every early bus which starts too early to be assigned to a split duty in the way described above, a set of split duties are formed starting at the earliest relief time at which split duties can start. Similarly, for every late bus finishing later than the latest finishing time for middle duties, a set of middle duties is formed finishing at a relief time before the end of the bus. This produces p extra subsets of duties, and the additional constraints:

$$\sum_{j=1}^{l_i} d_j \leq 1 \qquad i=n+1,\ldots,n+p$$

The method was applied to a few small schedules, and Manington reported encouraging results. However, the generation of possible duties is very restrictive, particularly in the way in which the last p sets of

duties, which both start and finish in the middle of a running board, are formed. It is rarely possible to say with any confidence when such a duty should start or finish before the schedule is compiled, so that in general choosing only from these p predefined sets would be unlikely to give good results.

2.4.3 HASTUS

HASTUS is a system developed at the University of Montreal which was described at the Leeds Workshop in 1980 [32] and in later papers by Rousseau, Blais and others ([19],[33],[34]).

The formulation involves the partitioning of each block into pieces of work (i,j) starting at time i and finishing at time j. A duty is composed of two pieces of work (i,j) and (k,h), either of which could be a null piece, giving a one-piece duty. The formulation could also be extended to three-piece duties. For each possible duty (i,j,k,h), x_{ijkh} is an integer variable with value 1 if the duty appears in the schedule, 0 otherwise. The cost of the duty is c_{ijkh}.

For each block l, T_l is the set of possible relief times, including the start and finish times. y^l_{ij} is a binary variable which has the value 1 if the piece (i,j) on block l is part of a duty in the schedule, and 0 otherwise.

The formulation is then:

$$\text{minimise} \quad \sum_{i,j,k,h} c_{ijkh} x_{ijkh}$$

$$\text{such that} \quad \sum_{k,h} x_{ijkh} + \sum_{k,h} x_{khij} - \sum_{l} y^l_{ij} = 0 \quad \text{for all } i,j \qquad \text{(i)}$$

$$\text{and} \quad \sum_{i \in T_l} y^l_{ik} - \sum_{j \in T_l} y^l_{kj} = b^l_k \quad \text{for all } k \in T_l, \text{ for all } l \qquad \text{(ii)}$$

$$\text{where} \quad b^l_k = \begin{cases} -1 & \text{if k is the starting time of block l;} \\ +1 & \text{if k is the finish time of block l;} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{and} \quad x_{ijkh} = 0 \text{ or } 1$$

$$y^l_{ij} = 0 \text{ or } 1.$$

The formulation may also include a small number of additional constraints on the duties selected; for instance, there may be a limit (upper or lower) on the number of one-piece duties.

The constraints (i) relate the partition of each block into pieces (i,j) to the duties selected for the schedule. The constraints (ii) correspond to the partitioning of each block into pieces, formulated as a network flow

problem.

A solution to this mathematical programming problem would lead to a feasible crew schedule. However, the problem is too large to solve except for very small schedules, and Rousseau et al. describe a simplified formulation using several relaxations of the original.

The first relaxation is that instead of using the actual relief times in each block, relief opportunities are assumed to occur at predetermined times, for instance on the hour and every 15 or 30 minutes. This considerably reduces the number of feasible pieces (i,j) and the number of possible distinct duties. Whereas in the original formulation, (i,j) is a feasible piece if there is a block with relief times i and j, now i and j must both be in the set of predetermined times, and then (i,j) is a feasible piece on every block with starting time at i or earlier and finishing time at j or later.

The second relaxation is that the duties selected must be sufficient to cover the total requirement of drivers in each time period, rather than covering each block individually.

This relaxation loses much of the structure of the original bus schedule, and to minimise the effect of this, some additional constraints are added. One set of these ensures that any short blocks which the user specifies should be covered by a single driver, are not broken. Another set of constraints ensures that the number of duties which have a piece of work starting at time t is at least equal to the number of blocks starting at time t.

The final relaxation is that the variables x_{ijkh} are allowed to be non-integer.

The new formulation is called the HASTUS-macro model. It can be solved using standard L.P. algorithms. Rousseau et al. claim that the solution is a lower bound to the solution to the original formulation, and that in practice it is a very good lower bound. HASTUS-macro by itself has been used in several places as a way of estimating the cost of proposed changes to the union agreement: normally, an accurate estimate requires actual crew schedules to be produced.

(However, situations in which the HASTUS-macro solution could be more expensive than the least cost schedule can be imagined. For instance, if the minimum mealbreak was much less than the next higher multiple of the length of the time period, the most efficient real duties would have no counterpart amongst the HASTUS-macro duties, which might mean that the HASTUS-macro schedule required more duties than

the original problem.)

For instance, Mitchell [35] compares the use of RUCUS and HASTUS-macro in estimating the cost of changes to the union agreement in the Los Angeles bus company, SCRTD. Estimating the effect of a single work-rule change by compiling a crew schedule for a representative division takes a considerable amount of scheduler time and effort, even when using RUCUS. Furthermore, because RUCUS is highly sensitive to changes in the scheduling rules, the estimation often requires changes to the logic of the program, and for some rules, the effort involved is not worthwhile unless the rule is actually implemented. The HASTUS-macro system is much more flexible in adjusting to different rules, and was found to be easier to use.

When an actual crew schedule is required, the solution of the HASTUS-macro model is used as an indication of the types of pieces of duty which are required. The transition from the HASTUS-macro solution to a feasible solution to the original problem is done in two stages: first the running boards are partitioned into pieces of duty, and then these are combined to form duties. This part of the system is called HASTUS-micro.

The partitioning of the bus schedule into pieces of work is done by solving heuristically a quadratic integer programming problem:

$$\text{minimise} \quad \sum_{i,j} \left[\sum_{k,h} \bar{x}_{ijkh} + \sum_{k,h} \bar{x}_{khij} - \sum_{l} y^{l}_{ij} \right]^2$$

where $\bar{x}$ is the HASTUS-macro optimal solution, subject to the constraints (ii) of the original formulation.

A solution is found by taking any feasible solution and fixing the values of the y variables for all the blocks except one; finding the values of y for the remaining block then becomes a shortest path problem. A new set of y values is found for each block in turn, and this is repeated until no further improvement in the objective is obtained.

Having partitioned the blocks into pieces of work, the pieces are formed into duties by solving a matching problem (although the resulting set of duties is allowed to include individual pieces of work, corresponding to trippers). This gives a feasible schedule. [34] describes a marginal improvement algorithm, in which each block is considered in turn and re-partitioned into pieces of duty. After each re-partition a new matching problem is solved, and the new schedule is kept if it is better than the old one. This algorithm was still under development.

Apart from the use of HASTUS-macro as a means of estimating the effect of changes to the union agreement, HASTUS has been implemented

as a crew scheduling package in Quebec and Montreal. The Quebec implementation, in 1979, was a batch system, but for Montreal an interactive system was developed, described by Rousseau and Blais [33]. They claim that "the HASTUS system left alone can produce a better solution than the best scheduler could normally produce. However, from the solution proposed by HASTUS, the good scheduler can obtain an even better solution."

In the Montreal system, the straight runs (duties without a mealbreak), are formed first. The scheduler examines them and if necessary edits them. They are then fixed, and the HASTUS system produces a complete schedule using HASTUS-macro and then HASTUS-micro. The scheduler again can modify the schedule by setting up files which describe the characteristics of desirable or undesirable duties: the schedule is then re-optimised. For a Montreal test schedule requiring about 280 drivers, the cost of the HASTUS schedule was 3% less that the manual schedule.

An advantage of the HASTUS system compared with the set partitioning and covering models is that very large schedules can be compiled. The number of constraints in the HASTUS-macro model depends on the time interval chosen, and to a small degree on the number of blocks in the bus schedule; the number of variables (possible duties) depends on the union agreement. Hence, the time taken to solve the HASTUS-macro model is hardly affected by the size of the schedule. The time taken to produce an actual schedule using the HASTUS-micro system depends on the number of blocks and on the number of spells of duty, so that the time taken increases with the size of the schedule, but by no means as fast as in the set partitioning and set covering models. In [19], the results of applying HASTUS to four test schedules are described: the schedules ranged in size from 28 blocks and 175 bus hours, to 287 blocks and over 2000 bus hours. Rousseau et al. reported that from the HASTUS point of view, all four problems were of approximately equivalent size.

In both Quebec and Montreal, the union agreement allows continuous duties without a mealbreak; compiling schedules in these circumstances is generally easier than under the normal U.K. conditions where all duties must work on at least two buses. However, HASTUS has been used to compile test schedules for at least four U.K. bus companies, and over the last eighteen months it has been bought by Tyne and Wear P.T.E., South Yorkshire P.T.E. and Lothian Region Transport.

Of the mathematical programming systems described in this

chapter, HASTUS appears to be the most successful, because it has been shown to be adaptable to different agreements, and because it can handle much larger problems than other systems.

(It is interesting to note that one of the earliest attempts to construct crew schedules by computer, reported by Young and Wilkinson [5] in 1966, used an aggregation method similar to the HASTUS-macro model, but failed to produce acceptable results.)

2.4.4 An Integrated Bus And Crew Scheduling System

Ball, Bodin and Dial [36] and Ball and Bodin [37] describe an algorithm which constructs a bus schedule and a crew schedule simultaneously, starting from a set of bus trips. The crew scheduling problem is formulated as a partitioning problem on an acyclic graph, in which each bus trip gives rise to two vertices, corresponding to the first part of the trip, from the start terminus to the relief point, and the second part, from the relief point to the end terminus.

The edges of the graph correspond to the possible actions of the crew at the beginning and end of a bus trip, and at the relief point; for instance, at the end of a bus trip, the crew can either return the bus to the garage, or drive it to the start terminus of another bus trip (which may be in the same location as the end terminus of the current trip, in which case the edge corresponds to a period of layover). Hence, the action of the crew at the end of a bus trip defines the linking of the trips into blocks, and so defines the bus schedule.

Another type of edge connects the two vertices corresponding to the two parts of a bus trip, and represents the crew staying on the bus past the relief point. There are also several types of edge corresponding to crews leaving one bus and joining another, or starting or finishing a duty.

There are two additional vertices, s and t, representing the garage; edges out of s represent a crew signing on and edges into t represent a crew signing off. Any legal duty corresponds to a feasible path from s to t, and a partition of the crew scheduling graph into a set of feasible paths gives a feasible solution to the crew scheduling problem.

A simpler graph with the same vertices can be defined for the bus scheduling problem, and a combined bus and crew schedule can in theory be found by partitioning the vertices in each graph into a set of feasible paths, subject to some restrictions to ensure that the two solutions are compatible. However, the algorithm described in [36] and [37] considers only the crew scheduling graph. The bus schedule is determined as a consequence of defining the crew schedule, as described above, but the

objective function is such that dead running time and layover time are penalised, to ensure that the bus schedule is reasonably efficient.

The crew scheduling problem is solved by first finding a set of spells of work rather than complete duties (a spell of work is a continuous piece of work on one bus). The edges corresponding to mealbreaks and joinups can therefore be dropped from the graph. The spells are constructed so as to minimise an objective function based on the total number of spells, as well as the dead running time and layover time in the implied bus schedule, and each spell is subject to a maximum length of say 4:30. The algorithm uses a heuristic method which builds up the spells incrementally, rather than solving the problem optimally. After an initial set of spells has been constructed, an improvement algorithm considers combining and resplitting pairs of spells and solves this as a matching problem; possible changes made at this stage may reduce the number of blocks in the bus schedule.

Finally, the spells are combined into a feasible set of duties. Duties can consist of between two and four spells, but in order to use a matching algorithm, pairs of short spells are first combined into partial duties. The complete duties are then formed by finding the minimum cost matching of the resulting pieces of duty, each piece being either a partial duty or a spell.

Ball et al. reported that this algorithm had been used on an experimental basis to compile a large schedule (consisting of nearly 1000 bus trips and more than 130 duties) from Baltimore. In this schedule, the number of splits duties and the number of trippers were limited. These additional constraints were handled by adding penalties to the costs of these types of duty. A number of different solutions were found, varying these penalties each time, to arrive at an acceptable schedule. The best schedule was slightly cheaper, in terms of paid hours, than the existing manual solution, but in some respects was not as good. No further work on this method was reported at the 1983 Workshop and it appears that it did not progress beyond the experimental stage.

The approach is interesting because, in theory, solving the bus and crew scheduling problems simultaneously could give better results than solving them separately. However, since the combined problem cannot be solved optimally, the theoretical advantage cannot be realised. In fact, it seems probable that using good bus and crew scheduling methods consecutively could produce better schedules than the algorithm described here. For instance, a bus scheduling system such as VAMPIRES [2] can produce a near-optimal bus schedule, and although Ball et al. do

not discuss the quality of the bus schedule produced by their algorithm, it seems unlikely to be as good. The decomposition of the crew scheduling problem into the selection of spells followed by the combination of the spells into duties is also likely to give poor results in many situations.

2.5 Conclusions

Originally, heuristic systems were intended to be run automatically and to produce implementable schedules without intervention from the scheduler. However, experience with both RUCUS and TRACS showed that a considerable amount of adaptation for each union agreement was necessary to produce good results.

The difficulty lies in the inflexibility of heuristic systems in constructing the initial schedule. The schedule is built up sequentially, one duty at a time, and the last few duties, when almost all the work has already been covered, may be very poor. The refining routines may remedy this to some extent, but it may be difficult to undo the effect of a wrong choice made early in the process, since all the subsequent duties will be affected.

The interactive elements which have been introduced into heuristic systems in recent years have been designed to improve the quality of the schedules by making use of the scheduler's expertise. The compilation of the schedule is under the control of the scheduler, who then has the responsibility for ensuring that a good schedule is produced.

Proponents of the RUCUS system claim that it produces cheaper schedules than manual methods, with an average saving of 1%. These claims apparently relate both to the original automatic version of RUCUS and to the more recent interactive versions, although since the quality of the schedules produced by interactive systems depends to a large extent on the skill of the scheduler, these claims may be hard to justify.

Because the construction of the initial schedule in heuristic systems is a simple sequential process, adding one duty at a time with no back-tracking, it takes little computer time. The refining routines may be relatively time-consuming, but overall, heuristic systems are very much quicker, and hence cheaper, to run than mathematical programming systems, and this is their main advantage. In some systems, such as RUCUS, it is suggested that because constructing schedules is so quick, the scheduler can produce several different versions by varying the values of the control parameters, and choose the best. However, because of the inherent problems of heuristic systems, there is no guarantee that the best schedule would be very good.

Of the mathematical programming systems which have been implemented, many appear to depend on the existence of a large proportion of one-bus duties, and it is doubtful whether they would otherwise produce good results. The set covering and partitioning models are also limited in the size of schedule which can be compiled; only one of the systems described here (section 2.3.4) attempts to solve larger problems by decomposing them into manageable subproblems, and that system has not been implemented. Potentially, however, set covering and partitioning models are easily adaptable to different union agreements, although none of the systems described here has been used by more than one bus company.

The HASTUS system appears to be the most successful of the mathematical programming systems, although it is not yet clear whether it can be adapted to produce results consistently as good as the manual schedules for the U.K. type of agreement.

All of the mathematical programming systems require much more computer time than heuristic systems, and hence are more expensive to run. This can be justified if they produce better schedules than manual or heuristic methods, or if they can produce comparable results with less scheduler effort.

2.6 Air Crew Scheduling

The air crew scheduling problem has many similarities to the bus crew scheduling problem. The equivalent of a piece of work on a bus, i.e. the smallest indivisible unit which must be worked by one crew, is a "flight leg", an individual flight from take-off to landing. Each duty, or "pairing", consists of a sequence of flight legs starting from and returning to the crew base; a pairing may include several duty periods alternating with rest periods. As with bus crew scheduling, the problem is to find a set of legal pairings which covers every flight leg and minimises the total cost.

There are a number of differences between bus crew scheduling and air crew scheduling, however, some of which affect the success of various solution methods. The planning horizon in air crew scheduling is usually a week or a month, rather than one day, as in bus crew scheduling, and there are no peaks in demand over the planning horizon as there are in weekday bus operations.

Because of the geographical spread of airline operations, there are few ways of linking one flight leg to another to form part of a pairing; in practice, the choice is usually restricted to flight legs departing from the

same point as the arrival of the previous leg, although the crew can make a "deadhead" flight as passengers to another point in order to pick up the next flight leg to be worked. In bus crew scheduling, on the other hand, even if there is more than one relief point, it is usually practicable to travel from any relief point to any other relief point, so that any piece of work in the bus schedule can be linked to any other piece to form part of a duty, provided that the interval gives a reasonable mealbreak or joinup.

Attempts to solve the air crew scheduling problem by computer have been based on a set partitioning or set covering model. The choice depends on the treatment of deadhead flights, in which the crew travel as passengers to connect with the next flight leg to be worked, or to return to the crew base at the end of the pairing. In a set covering formulation, deadheading is simply represented by over-cover, i.e. two or more crews assigned to the same flight leg. With a set partitioning formulation, pairings involving deadhead flights are explicitly generated.

Successful implementations of set partitioning and covering models arose much earlier in air crew scheduling than in bus crew scheduling. For instance, a survey paper by Arabeyre, Fearnley, Steiger and Teather in 1969 [38], reported on the progress already made by several commercial airlines. The limitations of I.L.P. codes at that time meant that in most cases, the problem size had to be severely restricted by a variety of means. In some cases, the number of constraints (corresponding to flight legs) and possible pairings was reduced by arbitrarily combining flight legs together and specifying that a combined flight leg should be treated as a single unit; in some cases, schedules were produced for one day at a time, and the combination of the pairings for each day into feasible pairings covering a week was treated as a separate problem.

Since then, attention has been concentrated on two approaches: better methods of finding exact solutions to larger set partitioning problems, and heuristic methods for solving set partitioning and set covering problems.

2.6.1 Rubin's Algorithm

This algorithm was described by Rubin in 1973 [39]. It is essentially a heuristic improvement algorithm which involves solving thousands of extremely small set partitioning problems. An initial solution is found by some means: Rubin suggests assigning a crew to each flight leg, or to a sequence of flight legs forming a duty period, and adding artificial instantaneous deadhead flights to and from a crew base to make a

pairing.

The pairings in the initial solution are sorted in descending order of cost. Subsets of up to n pairings are chosen, starting with the most expensive subsets. These n pairings cover a subset R of the total set of flight legs. All possible pairings covering those, and only those, flight legs in R are generated, and the minimum cost set of pairings covering R is found by solving a set partitioning problem. The minimum cost set of pairings replaces the original n pairings in the incumbent solution.

All possible subsets of up to n pairings are considered. The value of n is usually small (3, or rarely 4), and the search may be terminated early by a time limit: Rubin reported that for problems requiring more than 50 pairings in the solution, the processing of subsets of three pairings at a time would not normally be completed.

Rubin also reported that in 99% of cases, the original n pairings selected from the incumbent solution are the optimal solution to the set partitioning problem. Usually, matrix reduction and other techniques can demonstrate in advance that the incumbent subset is the optimal solution, so that the set partitioning problem need not be solved. However, because many thousands of pairing subsets are considered, substantial improvements can eventually be obtained, and in Rubin's test cases, improvements over the previously best known solutions of up to 55% were found.

The survey by Bodin, Golden, Assad and Ball in 1983 [40] reported that Rubin's algorithm became a standard approach in the airline industry, and that adaptations of it are still in use; it is very slow to run, but generally gives good results.

Rubin's algorithm would not produce good results for bus crew scheduling problems, however. The pieces of work in a spell of duty are a continuous period of driving time on one bus, whereas the flight legs in a pairing are much less closely connected. There is more scope for rearrangement of the flight legs from two or three pairings than if two or three bus crew duties are considered; the rearrangement then has to consider the spells of duty rather than individual pieces of work, and although improvements in a schedule of reasonable quality might be made by such rearrangements, they are not sufficient to produce a good schedule from scratch.

2.6.2 Marsten's Set Partitioning Algorithm

Marsten [41] developed an exact algorithm for the set partitioning problem which has been successfully applied to air crew scheduling

problems. It relies heavily on the logical deductions which can be made because each row must be covered exactly once: for instance, the choice of any column means that any other column covering one or more of the same rows can no longer be chosen. The method is therefore not applicable to set covering problems.

Marsten's approach involves finding an optimal solution to the continuous relaxation of the set partitioning problem in the usual way. When finding an integer solution from the continuous optimum, a branch-and-bound algorithm is used but instead of branching on an individual variable, the branching decision involves deciding which set of variables should be responsible for covering a particular row.

The columns of the constraint matrix $A = (a_{ij})$ of the set partitioning model are sorted into blocks B_i, where B_i is the set of columns which have their first 1 in row i (B_i may be empty for some i). Since each row must be covered exactly once, and every column of block B_i covers row i, any feasible partition can contain at most one column from each block.

The subproblems at a node in the branch-and-bound tree are formed by assigning the next unassigned row to one of the blocks which can cover it. However, although initially row r can be assigned to some or all of the blocks $B_1,.....,B_{r-1}$ as well as to block B_r, by the time that rows 1 to r-1 have been assigned to blocks, some of the previously eligible blocks can no longer be assigned to row r, because of the set partitioning constraints. This limits the number of subproblems to be formed at each node.

Using similar logical arguments based on the partitioning constraints, the addition of a new block to cover row r to an existing partial assignment covering rows 1 to r-1 can be used to eliminate many of the columns from the assigned blocks. If any of the eliminated columns are basic in the L.P. optimum solution given the assignment covering rows 1 to r-1, the solution is no longer feasible in the subproblem, and a new L.P. optimum is found.

This algorithm has been successfully applied to relatively small air crew scheduling problems, as described by Marsten and Shepardson [42] and Marsten, Muller and Killion [43]. The algorithm is being used by Flying Tiger, an air cargo carrier which has two fleets, consisting of six and eighteen aircraft, which are scheduled separately. Schedules covering a month are required, but for the smaller fleet, each week is identical so that the resulting scheduling problems are sufficiently small for the algorithm to be applied. For the 18-aircraft fleet, however, each

week is different and the monthly scheduling problem would have 300 constraints and 40,000 variables, which is too large to be solved by Marsten's algorithm. The algorithm has also been tested by Pacific Southwest Airways, on a set of small scheduling problems.

Many of the basic flight legs can be combined, because for practical reasons they must always appear together in a pairing: the number of constraints in the set partitioning problem resulting from this combination of flight legs is relatively small (mostly less than 150 in the examples quoted). It appears from the description in [43] that the number of pairings generated is limited by additional rules so that pairings with a low ratio of flying time to paid time are not formed.

In about two-thirds of the sample problems, the solution to the relaxed L.P. problem was integral, so that Marsten's branch-and-bound algorithm was not in fact required. Of the remaining cases, the smaller problems were solved to optimality; the larger problems were eventually terminated and the incumbent integer solution (if any) accepted.

Gerbracht [44] describes a variant of this algorithm in which Marsten's branching strategy is used, but instead of computing bounds by linear programming, Lagrangean relaxation and subgradient optimisation are used. Gerbracht compared this algorithm with Marsten's on a set of sample problems from Continental Airlines and reported that the new version was somewhat faster. The problems were of similar size to the Flying Tiger and Pacific Southwest Airways problems.

For the tests on the Pacific Southwest Airways problems, manual solutions were also produced, and Marsten's algorithm usually showed savings of 15-25% over the manual schedules. Flying Tiger are reported in [43] to be using the system regularly to schedule their smaller fleet, with direct cost savings of $300,000 annually. For scheduling problems which are sufficiently small, this seems to be a successful approach, and Marsten & Shepardson suggest that in order to make use of it for larger problems, it would be worthwhile to investigate heuristic decomposition methods, so that the problems can be broken into subproblems which are small enough to be solved exactly.

Although the algorithm is reported to be successful for small air crew scheduling problems, it would not be applicable to bus crew scheduling problems. As discussed in chapter 6, the set covering formulation is more appropriate for bus crew scheduling; if a set partitioning formulation were used, far more candidate duties would have to be generated from which to select the schedule, to compensate for the fact that once a duty has been selected, no other duty covering the same

pieces of work is eligible.

2.6.3 Heuristic Methods

For very large set partitioning and set covering problems, exact algorithms are impracticable. A number of papers by Baker and others ([45],[46],[47]) describe heuristic crew scheduling methods developed for the Federal Express Corporation, a U.S. airline specialising in the overnight delivery of small-scale freight. A set covering formulation of the crew scheduling problem is used, but as the problems are too large to solve exactly, heuristic solution methods are used instead.

A set of feasible pairings, which form the variables of the set covering problem, is first generated. One of a number of heuristic algorithms is used to select a solution from this set; this initial solution is then improved, also by heuristic algorithms.

The heuristic algorithms which were investigated for constructing the initial solution are also described by Bodin, Golden, Assad and Ball [40]. Each of the four algorithms begins with an empty solution set and adds pairings one at a time until every flight leg is covered; the criteria for choosing the next pairing are:

A. add the pairing for which the ratio of cost to the number of so far uncovered flight leg is minimised;

B. sort the pairings in ascending order of cost per flight leg; add the next pairing in order which contains no previously covered flight leg. If a complete pass through all the pairings still leaves uncovered flight legs, repeat, allowing pairings with one more previously covered flight leg on each pass until the solution is complete;

C. add the pairing with the maximum number of uncovered flight legs;

D. add the pairing which minimises the ratio of covered to uncovered flight legs.

Experiments with data from Federal Express indicated that Algorithm A is superior to the others for air crew scheduling problems. In [47] the results obtained using this algorithm are compared with the optimum solution of the L.P. relaxation for nine set covering problems, small enough for the L.P. to be solved. For six of the problems, the L.P. optimum was in fact integer, so that the solution was optimal for the original problem. In each case, Algorithm A was used to produce ten different solutions, of which the best was chosen; it is not clear how the different solutions were obtained. The deviations between the L.P. optima and the best heuristic solutions were relatively small, with a maximum of

18.8% for an example in which the L.P. solution was fractional; in two cases, the heuristic method found the optimal solution.

Baker, Bodin, Finnegan and Ponder [45] describe an improvement heuristic which they term an extended 2-opt procedure. If two pairings have a common point, i.e. the crews land at the same airport at some time in each pairing, it may be possible to reduce the total cost by splitting the pairings into two parts at that point and making an exchange between the two pairings. (The equivalent of this in bus crew scheduling terms is a commonly-used improvement heuristic in systems such as TRACS and RUCUS.) The extension of this described in [45] allows for K pairings with a common point, and solves a K×K assignment problem to find the minimum cost set of pairings.

Baker [47] describes a solution merging technique which takes two candidate solutions and merges them to form a cheaper composite solution; the composite solution is guaranteed to be no more expensive than the cheaper of the two original solutions.

The algorithm identifies sets of columns from the two candidate solutions which cover mutually exclusive subsets of the rows. To start a new row subset, an unassigned column from either solution is chosen, and the rows covered by it form the row subset. The subset is augmented by looking for unassigned columns in either solution which cover any of the rows in the current subset. (Usually, there will be several columns in the other solution which cover rows of the initial subset, but no other columns in the first solution, unless there is over-cover.) The subset is complete when there are no remaining unassigned columns which cover any rows in the subset. When all the columns from both solutions have been assigned to a set, then for each row subset, the cheaper of the two column sets covering it is selected for the composite solution.

Usually, the final merged solution was cheaper than any of the original solutions, although the improvement was usually very modest (about half of one per cent), mainly because the extended 2-opt procedure mentioned above had already been applied to most of the problems.

The combination of heuristics used is reported to produce schedules of operational quality for Federal Express. It can produce solutions for very large set covering problems, with thousands of rows, in a fairly modest amount of computer time. However, Baker and his colleagues suggest that these scheduling problems are untypical of air-crew scheduling in general, and that the heuristics may not be applicable to passenger air-crew scheduling.

The good results reported by Baker et al. for these heuristics have not been found when the methods have been applied to bus crew scheduling problems. The method used in IMPACS for constructing a starting solution, as described in section 5.3.2, is similar to Algorithm A. Several different algorithms have been tested, including those described above, and none proved itself consistently better than the others. The IMPACS starting solution always has several more duties than the final solution, and there is always a considerable amount of over-cover, so that the heuristic performs far worse than it does for the Federal Express problems.

The solution merging technique has also been tested by the author on the bus crew scheduling problem as a possible means of producing improved starting solutions in IMPACS.

It has invariably been found, however, that unless the two solutions being merged have a column in common, in which case the rows covered by that column form one of the required mutually exclusive subsets, the rows cannot be divided into subsets. The procedure for augmenting the first row subset does not terminate until all the rows have been included. Hence, the composite solution is always identical to the cheaper of the two initial solutions. Again, the merging technique gives much better results when applied to air crew scheduling problems.

2.6.4 Air Crew Scheduling - Summary

For sufficiently small air crew scheduling problems, Marsten's exact set partitioning algorithm works well, and can produce considerable savings in operating costs compared with manual methods. The method is also applicable to set partitioning problems in general, provided that they are of the right size. Larger problems can only be solved by heuristic methods; Rubin and Baker have developed two approaches which have been successfully used. However, for the reasons given earlier, they do not appear to be applicable to bus crew scheduling problems; despite the similarities between the two types of problem, in practice, successful solution methods are not transferable from one to the other.

2.7 Rail Crew Scheduling

Little successful work has been done on the computer scheduling of train crews, although commuter rail services, at least, appear to have many similarities to urban bus operations.

A potential difficulty which arises in rail crew scheduling which does not occur in bus crew scheduling is that often the crews are based at

many different points throughout the rail network, with a fixed allocation of crews at each point. It is known from discussions on the possible applications of the IMPACS system that this happens on the Melbourne local rail network, and on the London Underground services, as well as in British Rail. Although a bus schedule may conceivably be operated by crews from more than one depot, in practice this is rare, and the number of depots involved would be small.

When crews are based at more than one point, duties must sign on and sign off at the same point. It may also be necessary for the crew to travel from their home base to the starting point of their first piece of work, or from the point where they leave the last train worked back to base. In that case, there could be several different duties all covering exactly the same work but assigned to different bases, and with different costs because of variations in travel time. The crew schedule must also have the correct number of duties assigned to each crew base. These additional complexities may make rail crew scheduling less amenable to the development of computer methods than bus crew scheduling. For instance, if a set covering or set partitioning model is considered, the limits on the number of duties to be assigned to each crew base will increase the number of constraints and may make the problem too large to solve.

Ellis and Savin [48] describe the computer scheduling systems developed by British Rail. Although some work has been done on mathematical programming methods, this approach was later abandoned, and the systems currently in use or under development are based on interactive methods. The main types of schedule involved are the long distance schedules and local services schedules, where the trains work much shorter distances within a small area, for instance operating commuter services.

Interactive systems have been developed for both types of schedule. Long distance schedules are relatively simple, since the crews can only work on a small number of trains during a duty, often just a return trip from their home base. The scheduler can use a number of commands to build up a schedule, some of which involve solving assignment problems to produce possible sets of duties for very small parts of the schedule.

Local services schedules operate over a core area, containing all the crew bases and relief points, and a periphery; trains run into the periphery, but crews cannot be changed there. Although the peripheries of local areas may overlap, for scheduling purposes the areas can be considered to be independent. The scheduling system developed for local

services uses colour graphics to represent trains on a VDU screen. The scheduler builds up duties one at a time; they are checked by the system and the work covered is removed from the display.

When there is only one crew depot, or a very small number of crew bases, and the difficulties mentioned above do not arise, there seems no reason why a bus crew scheduling system could not be easily adapted to local rail crew scheduling, or vice versa. For instance, the DIGOS system described in section 2.3.2 was originally developed for Dutch Railways to schedule guards, although no details are available of its operation in that form.

Tykulsker, O'Neil, Ceder and Sheffi [49] describe a rail crew scheduling system which was based on the bus crew scheduling methods presented at the Leeds Workshop in 1980. The system was developed for the New Jersey Transit Corporation, which operates a commuter rail network, as a means of evaluating the costs of changes to the work rules. It appears that crews could be based at more than one point, since it was necessary to ensure that each duty started and finished at the same point, but limits on the number of duties at each point are not mentioned.

Tykulsker et al. used a set covering model and found that over-cover was a common occurrence. Over-cover can represent deadheading, i.e. a crew travelling as passengers to pick up the next train to be worked, but the solutions often contained deadheading loops, in which a duty is assigned to a sequence of trains, starting and finishing at the same point, such that each train is already fully covered by another duty or duties. The loop is redundant and a new duty, generally cheaper, should be formed without the loop.

The solution method adopted for the set covering problem was to solve the relaxed L.P., in which the variables are allowed to take non-integer values, and form additional duties by removing deadheading loops from the duties in the non-integer solution. Then a new set covering problem was solved using only the duties from the incumbent non-integer solution, together with the additional duties, this time finding an integer solution. This method could not normally be expected to give good results, since it assumes that an acceptable integer solution exists which is close to the relaxed L.P. solution. However, Tykulsker et al. reported that good results were obtained and that the gap between the non-integer and integer objective functions was small. This may have been because the initial set of possible duties was restricted in such a way that the solution to the relaxed L.P. could be expected to be integer or nearly so,

according to the arguments of Ryan and Foster (section 2.3.5). The fact that such a restricted set of duties gave good crew schedules may have been because the possible duties were inherently restricted by the work rules, and also because it appears that duties without mealbreaks were allowed, which would simplify the scheduling task.

CHAPTER 3. THE IMPACS SYSTEM

3.1 Background

This chapter discusses the origins of the work done by the author on mathematical programming methods for bus crew scheduling, and the progress of the work is briefly described. The approach proved to be successful and eventually led to the development of a system suitable for use by schedulers. The system consists of a package of Fortran programs, and was named IMPACS (Integer Mathematical Programming for Automatic Crew Scheduling). The elements of IMPACS are summarised in section 3.3 and described in detail in the following chapters.

The work which led to the development of the IMPACS system began in 1978 as part of a broader study of several different aspects of computer applications in bus scheduling and routeing and bus crew scheduling, supported by an SRC research grant, which ran from 1977 to 1980.

At that time, a great deal of work was being done to try to improve the heuristic crew scheduling system TRACS, described in section 2.2.2, which had been developed at Leeds over a number of years. Experience had shown that the quality of the final result produced by TRACS depended very much on the quality of the initial solution; although the improvement techniques which were subsequently applied could modify the schedule considerably, it could be difficult to alter the basic form of the schedule laid down already. Unfortunately, the program which compiled the initial solution usually needed some modification before it could produce good results in a new situation, and sometimes an entirely new approach was needed. Hence, although it had originally been hoped that almost any method for forming an initial solution would be adequate, since the improvement routines, if sufficiently powerful, would be able to produce a good quality final schedule, it now seemed that in order to produce good schedules, a variety of techniques should be available for constructing the initial solution, and the appropriate technique or techniques selected according to the circumstances.

The program to produce the initial schedule had been modified so that it could be run in a number of different ways and it was intended that future work on TRACS should be devoted to deciding which combinations of techniques should be used in given circumstances. Some work had already been done on preliminary analysis of the bus schedule with the intention that this would provide the information required to choose the appropriate method for constructing the initial solution, the selection being done either automatically or by the scheduler. Even so, with each

new schedule there would still be the possibility that none of the existing techniques would be appropriate, so that a considerable amount of work might be necessary to develop a new approach.

It was thought worthwhile to investigate mathematical programming methods because of the difficulties which had been encountered with heuristic methods. It was clear that an exact solution to a mathematical programming formulation would in most cases be impossible to achieve, but in the report on work carried out under a previous SRC grant, it was stated that, "in certain cases there may exist such a small number of alternative duties that the best way to tackle the problem may be by use of integer linear programming methods." The intention therefore was that mathematical programming should be one of the techniques available within TRACS for constructing the initial solution, to be selected only in appropriate circumstances.

The work already done by Manington [31] as described in section 2.4.2, had used a formulation similar to set covering, in that a large set of possible duties was formed from which the schedule was selected. This work had shown that it would rarely be practicable to form all possible duties, so that it would be necessary to restrict the number of duties formed, and hence the more the generation of duties was restricted naturally by the scheduling conditions, the closer the optimal solution to the mathematical programming problem would be to the optimal schedule. Conversely, the heuristic methods often perform worse in restricted conditions, because there are fewer ways of constructing a good schedule and so the program is more likely to make a choice which will lead to a poor quality result. Hence, it seemed that mathematical programming and heuristic methods might be complementary, each making up for the other's deficiencies. It was felt that in spite of the evident drawbacks of heuristic systems, an alternative approach based solely on mathematical programming methods would only be successful for a limited range of problems.

The problem of lack of portability arises in heuristic systems because if the scheduling conditions are different from those previously encountered, e.g. the peaks in the bus schedule are more pronounced, or the union agreement contains new rules, the existing methods for building up an initial solution may no longer give good results: in just the same way, manual schedulers from different bus companies tackle crew scheduling in different ways, because they have adapted their working methods to the local conditions. For instance, in many cases, the best results are found by constructing early duties first, but if the conditions

make it particularly difficult to schedule the late duties efficiently, it may be better to start with those. Because heuristic methods, like manual methods, build up the initial solution incrementally, they have to be adapted to the particular circumstances in a similar way.

On the other hand, mathematical programming methods using a set covering formulation, or any similar formulation in which the schedule is selected from a large set of possible candidate duties, are not affected by changes in the scheduling conditions. Changes in the agreement will only affect the type of duty which can be included in the set of candidates and the selection method is invariant. Changes in the pattern of bus provision during the day have no effect on the method at all: the duties selected automatically fit the bus schedule, whatever its shape.

Hence, although it was thought that mathematical programming methods would only be successful for problems of limited size, it was felt that within that range the portability problems found with heuristic methods should not arise. If the range of mathematical programming methods could be extended to cover a significant proportion of schedules, they would give a much better basis for building a robust general-purpose crew scheduling system.

3.2 History Of IMPACS Development

As described above, the inital development work on mathematical programming methods was done under an SRC grant from 1978 to 1980. Although this was based on the work done by Manington, a simpler formulation was used. The new formulation, still used in IMPACS, is a variant of set covering in which both under- and over-cover are allowed, with appropriate costs in each case. Under-cover represents a piece of work not allocated to any duty; it usually incurs a very high cost and is not acceptable. Over-cover represents a piece of work allocated to two or more duties, and although it must be eliminated to arrive at a feasible schedule, it is often preferable to remove it by editing the solution rather than attempting to prevent it occurring.

Because it was important that good quality schedules should be available for the sample problems used in the early development work, for comparison with the results produced by the new system, the sample problems chosen were from Leeds district of West Yorkshire P.T.E.; since a considerable amount of work had been done to tailor TRACS to the requirements of the P.T.E., good TRACS solutions could be found for these problems. Although it had been anticipated that mathematical programming methods would only be appropriate in situations where the

scheduling conditions were very restrictive, this was not the case in the sample problems, and so artificial restrictions were imposed on the generation of duties in addition to those required by the union agreement. However, it soon became apparent that mathematical programming methods would be able to cope with a much wider range of conditions than had been anticipated, and that by using carefully-chosen additional restrictions, it might be possible to build a general-purpose scheduling system producing good quality results.

During the first phase of the development, the only mathematical programming package available was the XDLA package on the ICL 1906A, which was the University's mainframe at that time. The XDLA package proved to be slow and difficult to use, and the early work concentrated on the earlier duty generation stages of the system. Fortunately, in the beginning, the solutions to the non-integer relaxation of the set covering formulation turned out to be almost always integer or nearly so, and little effort was involved, if any, in deriving an integer solution. In fact, for development purposes, a simple rounding heuristic often proved satisfactory. Later experience showed that the high incidence of integer solutions was due to the fact that the problems were severely constrained by the additional restrictions placed on the duties generated; as the artificial restrictions were relaxed to allow better solutions to be produced, the difficulty in deriving an integer solution increased.

Even when run in non-integer mode, the XDLA package could only be run at night, and in integer mode it was extremely slow. Hence, when the first grant came to an end in 1980, it was clear that a better mathematical programming package would have to be found.

The initial work under this grant is described in [50]. It proved very promising, and when the grant expired in November 1980, SRC gave a further two-year research grant for an investigation devoted solely to developing this approach. At the same time, a Senior Visiting Fellowship for the first six months of 1981 enabled Dr. D.M. Ryan of Auckland University, New Zealand, to visit the University of Leeds. He provided a set covering package, ZIP, described in [51], which was specially designed for crew scheduling applications, thus solving the problem of finding a good mathematical programming package.

By the time the second grant expired at the end of 1982, a prototype of the IMPACS system had been developed; it had been shown to be easily adaptable to different scheduling agreements, and trial schedules for several bus companies had been compiled. A few large bus operators were at that time interested in sponsoring further development of the system,

and eventually a contract was agreed with London Transport for a programme of development lasting until December 1984. By that time the IMPACS programs had been extensively modified to make them easy for schedulers to use, and the programs could cope with all the commonly occuring features of London Transport's bus operations, so that almost all the schedules of London Buses Ltd. can now be compiled using IMPACS.

3.3 Outline Of IMPACS

In summary, the process of compiling a schedule using IMPACS involves first generating a large set of valid duties; from these, the duties forming the schedule are selected by formulating the scheduling problem as a set covering problem. Because the total number of valid duties is extremely large, it would be impossible to consider them all, and so the number of candidate duties generated is restricted in various ways.

In order to produce a crew schedule using the IMPACS system, the scheduler goes through the following steps:

1. Set up data and parameter files (chapter 9 and Appendix). Parameter values define the criteria to be met by the duties generated in step 3. The scheduler can set the parameter values to exclude many valid but inefficient duties and so limit the number of duties to be considered.

2. Run the relief time selection program to reduce the number of pieces of work to be covered (chapter 4).

3. Generate a large set of possible duties (chapter 5).

4. Reduce the number of duties by running the COMPARE program, if appropriate (section 5.11.2), and the EVEN program (section 5.11.3).

5. Run the ZIP program, which:

   a) formulates the scheduling problem as a set covering problem (chapter 6);

   b) relaxes the integrality constraints and solves the continuous L.P. (chapter 6);

   c) unless the solution to the L.P. is integer, looks for an integer solution using branch-and-bound (chapter 7). Search stops as soon as a sufficiently good solution is found.

6. Run the printing program, (chapter 8) which:

   a) translates the integer solution into a list of duties;

   b) applies heuristic improvement techniques;

   c) prints the schedule.

7. Edit the duties manually if necessary (chapter 8).

CHAPTER 4. RELIEF TIME SELECTION

4.1 Introduction

As already mentioned in section 2.3, set covering and set partitioning problems with more than a few hundred constraints are in practice impossible to solve within a reasonable time. The set covering model as used in IMPACS can therefore only be successful as a way of solving crew scheduling problems if the number of constraints is sufficiently small.

The interval between two relief times on a running board is a piece of work which must be assigned to one crew, and the pieces of work in the bus schedule correspond to the constraints of the set covering problem. Hence, the number of constraints in the set covering problem is at first glance determined by the number of relief times in the bus schedule.

However, if a relief time is not used as a changeover time by any of the duties which form the variables of the set covering problem, i.e. no duty starts a spell at that time, and no duty finishes a spell at that time, then the relief time can be treated as if it did not exist. The pieces of work either side of the relief time can be combined into one; in terms of the set covering formulation, the corresponding constraints are identical, so that one is redundant and can be dropped. So in fact, the constraints of the set covering problem correspond to the pieces of work defined by the relief times actually used by the duties generated, which will not necessarily be the same as the original pieces of work, defined by all the relief times in the bus schedule.

A few relief times cannot in any case be used by the duty generation process; for instance, relief times within the minimum spell length of the start or end of a bus will not be used, nor will relief times in the evening which are too late to be the end of a mealbreak. The number of constraints in the set covering problem can be further reduced if the duty generation process is only allowed to use certain specified relief times. For every relief time banned, the number of constraints is reduced by one. A heuristic procedure which analyses the bus schedule and selects certain relief times has been developed; any relief time not selected is ignored when duties are being generated.

As well as reducing the number of constraints in the set covering problem, rejecting relief times in this way limits the number of possible duties which can be formed, so that the number of variables is reduced at the same time.

The current limits in IMPACS on the size of the set covering problem

which can be solved by the ZIP program are 350 constraints and 5000-6000 variables. Since the limits are defined by array sizes they are fairly arbitrary, but increasing the size of problems beyond these limits would lead to a rapid increase in the computer time required to solve the problem, and might also lead to difficulties in finding an integer solution in any reasonable time.

If there were no selection of relief times, the 350-constraint limit could easily be reached by fairly small schedules in urban bus operations. The average interval between relief times on a radial bus route operating from a city centre to an outer terminus, passing the depot in each direction, might be about 30 to 45 minutes. A schedule requiring 50 duties, with an average driving time of say 6:30, would then have about 430 to 650 pieces of work.

4.2 Possible Risks

Although in many cases reducing the number of eligible relief times is essential in order to be able to solve the resulting set covering problem, there might be a risk that, in introducing a heuristic procedure which depends on an analysis of the bus schedule, the problems found in heuristic crew scheduling systems such as TRACS and RUCUS when applying the system to new conditions could arise. As described in chapter 2, the quality of the solutions produced by systems such as these is often dependent on the scheduling conditions, and they may require considerable modification before they can produce good results under a new set of scheduling rules or a new pattern of bus operation.

However, there is much less danger of this happening with the selection procedure described below than in heuristic systems, because the individual choices made have much less influence on the quality of the final result. As heuristic systems build up an initial schedule, the duties are chosen one at a time and each choice limits the way in which the rest of the work can be covered; although the improvement techniques applied to the initial schedule can change it considerably, it may be difficult to recover from poor choices made at an early stage, and hence it is important that the construction of the initial schedule should be correctly tuned.

On the other hand, a choice of a relief time by the selection procedure makes no commitment as to which duties will finally change over at that time, or even in most cases whether the relief time will be used at all. The procedure is simply intended to provide a reasonable set of relief opportunities.

There is a possibility, however, that there may be one or more critical relief times which must be used in order to produce a good schedule, and that the selection procedure may fail to identify these critical times. However, the larger the schedule and the more frequent the relief times are, the more likely it is that there are several alternative ways of covering the work, all of similar quality, and therefore critical relief times are less likely to occur in the large problems for which the procedure was principally designed. In any case, since the selection procedure is based on the construction of potential duties, as described below, any critical times which do exist should be correctly selected.

Since one of the original aims in developing IMPACS was to avoid the problems of transportability which had been found with the TRACS system, it was recognised that in view of the possible risks involved in the selection procedure, it should be tried out on many different schedules with different scheduling rules, different patterns of operation and so on. Otherwise, it is easy to allow the heuristic procedures to depend on implicit assumptions which are not generally true, for instance that split duties always cover both peaks.

Although there are several other heuristic techniques used at various stages in IMPACS to reduce the difficulty of solving the set covering problem, the others are based on the characteristics of the set covering problem, rather than the bus schedule, and problems of transportability do not arise.

4.3 Forward And Backward Marked Times

The first stage in selecting relief times is to mark certain relief times on each bus which are approximately the length of a spell apart, a spell being a continuous period of work on one bus.

(The procedure is described in terms of the maximum spell length, but in fact the spell length used is specified by the scheduler and sometimes, as discussed below, a shorter spell length may be appropriate.)

The maximum spell length, or maximum continuous driving time, may be specified in the union agreement, but it is more usual to specify the maximum stretch length, which includes various allowances, such as signing on or off, and these must be subtracted to find the maximum length of a spell.

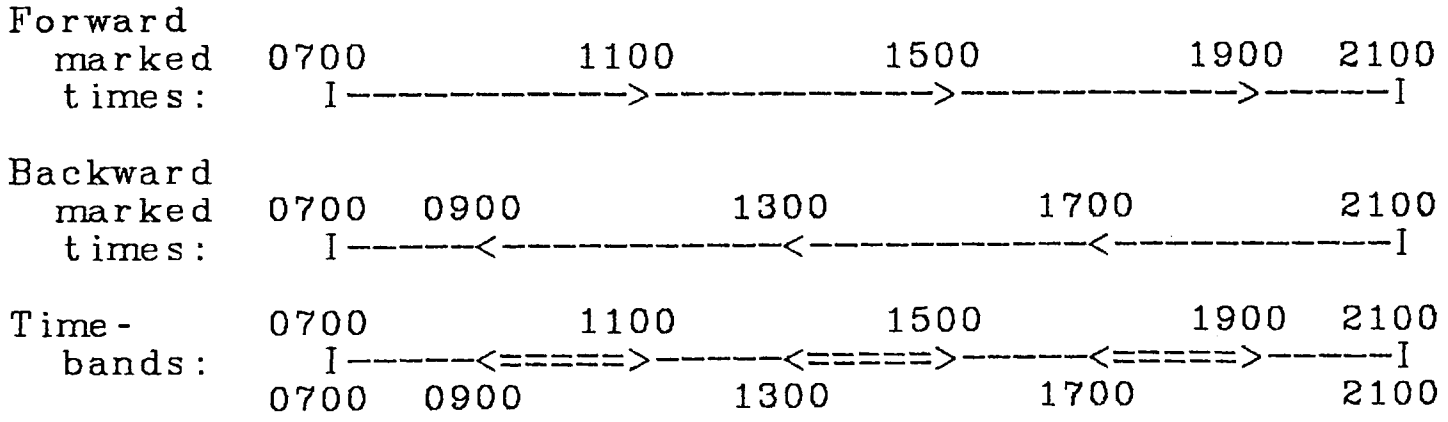
For each bus in turn, it is supposed that a crew starts work when the bus leaves the garage, and the latest time at which the crew may

leave the bus is marked. Then the latest finishing time on this bus for another crew taking it over at the marked time is also marked, and so on, until the bus returns to the garage. If crews on this bus are changed only at the marked relief times, then the bus will be covered by the minimum number of spells of work.

Similarly, relief times can be marked in the other direction, starting when the bus returns to the garage and working backwards in time until it first leaves the garage. If these "backward" marked times are used to change crews then, again, the bus will be covered by the minimum number of spells.

Sometimes the "forward" and "backward" marked times coincide, but generally each backward marked time is earlier than the corresponding forward marked time. Each pair of marked times defines a time-band within which crews should be changed over, if each bus is to be covered by the minimum number of spells.

For example, suppose a bus leaves the garage at 0700 and returns at 2100, with relief times every hour, and the maximum spell length is four and a half hours:



The bus can be covered by four spells of work, but not if any of the relief times at 0800, 1200, 1600 and 2000 are used.

In some schedules the maximum spell length may be much greater than can normally be achieved: for instance, in one case the maximum was 5:15, but the maximum spreadover of a duty, from signing on to signing off, was 8 hours, including the mealbreak, and signing on and off allowances, so that the average spell length was less than 3:30. It would then only be possible to have a spell of duty as long as 5:15 by making the other stretch very short, and it would be sensible to use a smaller value than 5:15 for the spell length used to determine the time-bands. In these cases, the argument for selecting times within the time-bands is rather different, since it cannot be claimed that these times must be used in order to cover the bus in the minimum number of spells, or that this is

the ideal to aim for anyway. In practice, however, it still appears that the time-bands are a good basis for selecting relief times.

When calculating the forward and backward marked times, it is possible to ban relief times on all buses during specified periods, for instance during the peaks or before the canteen opening time. The boundaries of the time-bands may then have to be moved away from where they would otherwise fall: forward marked times are moved earlier, to before the start of the banned period; backward marked times are moved later, to after the end of the banned period. To cope with cases where the driving time on some routes is restricted, a different spell length can be used on specified buses.

If there are pairs of relief times which are very close together (defined in the selection program as less than 20 minutes apart) and the scheduler so chooses, the program will attempt to select at most one relief time from each pair. This is taken account of when the time-bands are being marked, and throughout the rest of the program.

At the end of this stage, all relief times which fall within the bands defined by pairs of marked times are temporarily marked as being the most suitable candidates for selection. The start and finish of each running board are also marked, since these must be selected. All the rest are rejected, at least for the time being.

For example, Figure 4.1 shows the bus graph of a sample schedule. The relief times which fall into the time-bands calculated using a maximum spell length of 4:36 are shown in Figure 4.2. (Relief times after 2120 have been banned, since no middle duty can finish after that time.)

4.4 Further Analysis

Although the time-bands marked so far may exclude many of the original relief times, it is desirable to go further and reject those times within the time-bands which seem least likely to contribute to a good schedule. Briefly, this is done by constructing notional early and late duties which are initially restricted to the marked relief times: any relief times within the time-bands which are not used by these duties may be rejected. The analysis may also identify cases where it is necessary to choose relief times outside the time-bands in order to construct particular duties. Hence, the final selection of relief times, although based on the time-bands, will not include all the times within the time-bands, and may include a few times outside the bands. The construction of the notional early and late duties is described in the following sections.

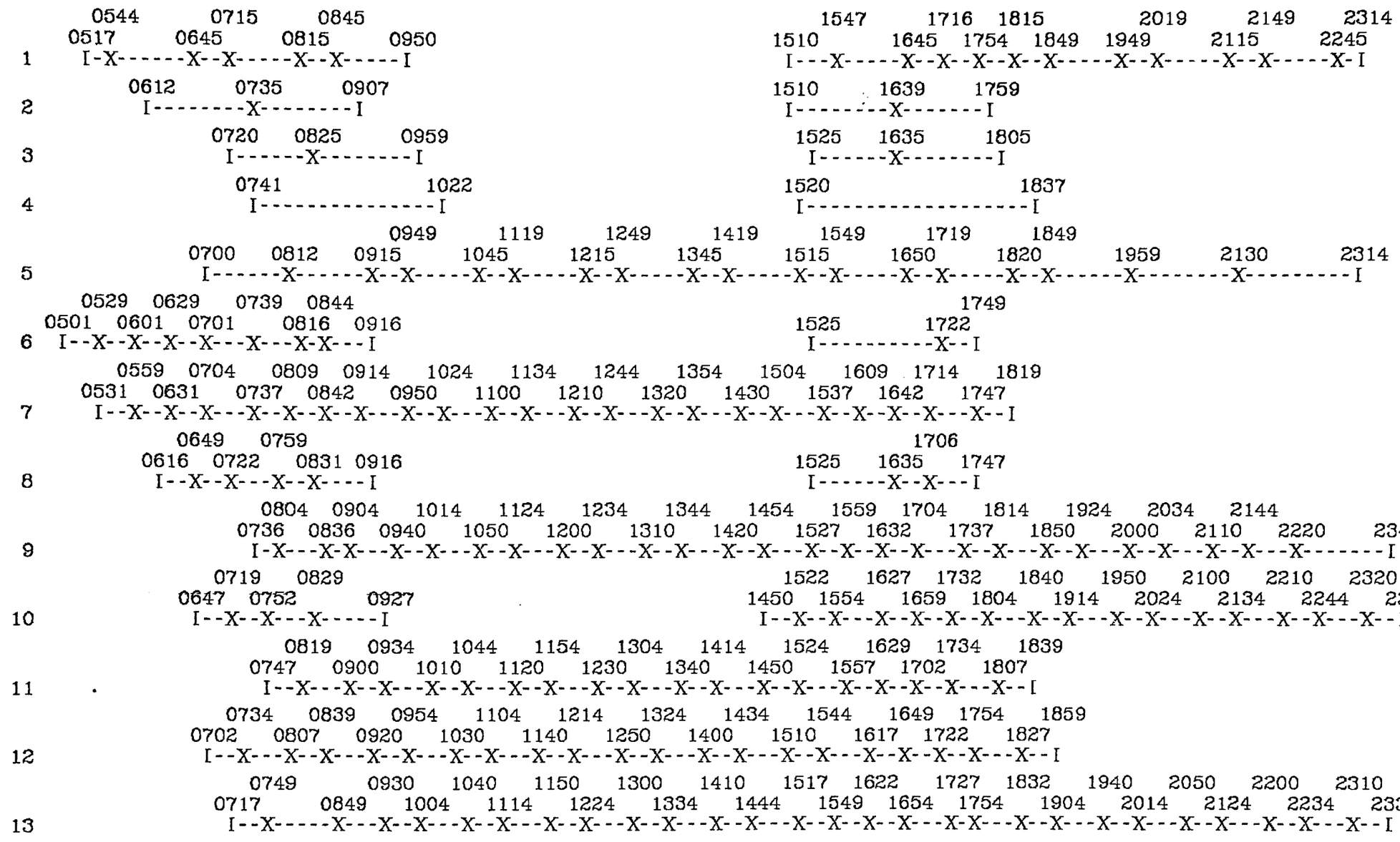


Figure 4.1 Sample Schedule To Show Selection Of Relief Times

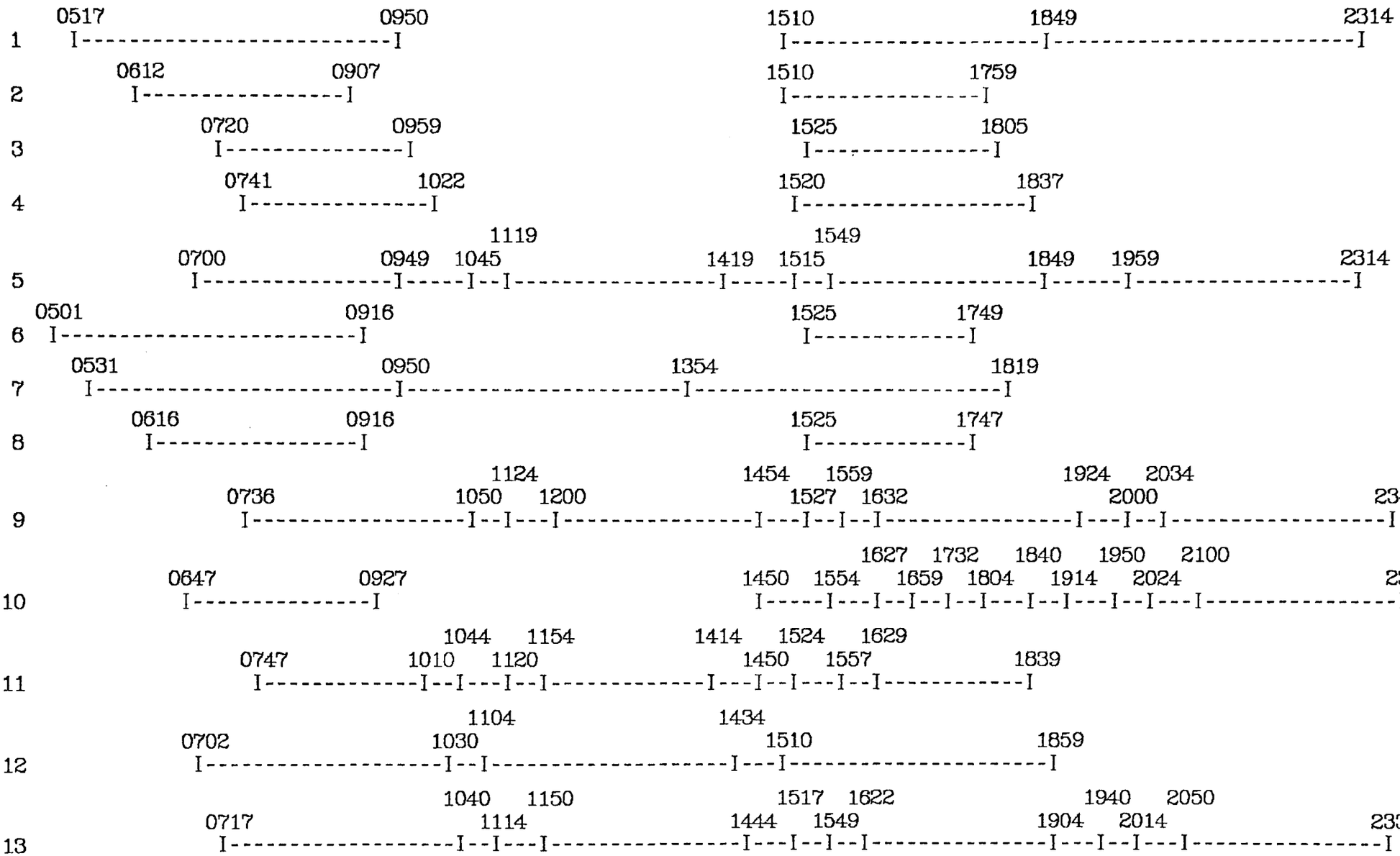


Figure 4.2 Time Bands For A Sample Schedule

4.5 Early Duties

The early duties which can be formed using the marked relief times are considered. An early duty starts by taking an early bus (that is, a bus which leaves the garage no later than the latest starting time for early duties) out of the garage, and normally works on that bus until the end of the morning peak. All the early buses, excluding those which return to the garage immediately after the peak, are examined to see whether they can be taken over at a relief time within the first time-band, by a crew which has already worked a first stretch of duty, so that the bus can be worked by the first stretch of another duty up to the changeover time.

For instance, if a bus starts at 0700 and the first time-band is 0949 to 1119, then ideally, the bus should be worked by the first stretch of one duty from 0700 up to a relief time in the range 0949-1119, and then be taken over by a crew starting the second stretch of another early duty.

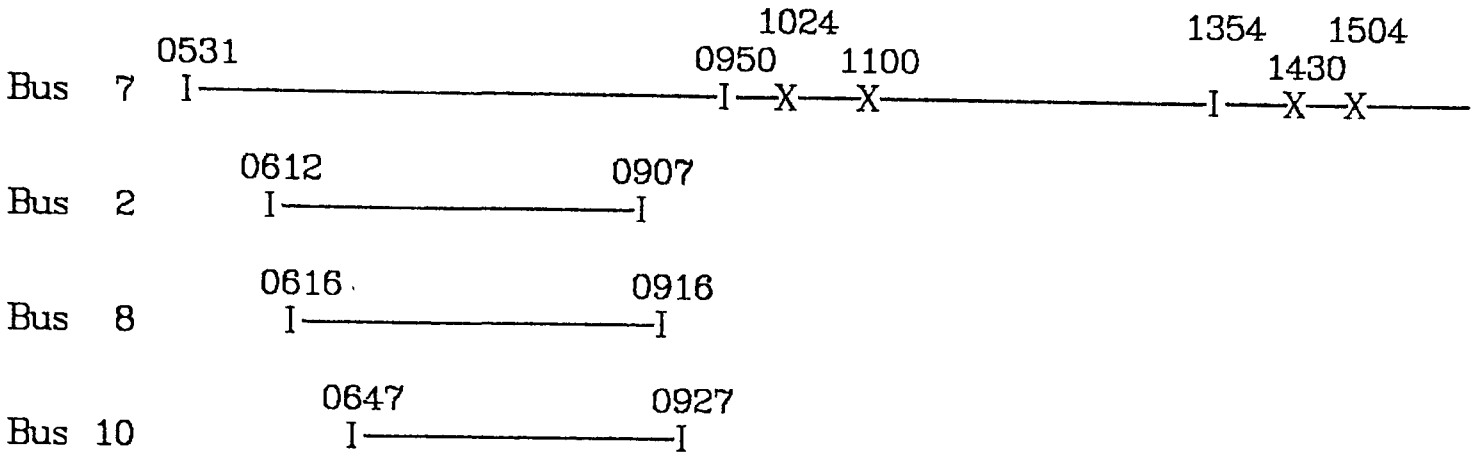
If the early bus can be taken over by a sufficient number of early duties in this way (where "a sufficient number" is set arbitrarily at four), any relief time within the first time-band at which it cannot be taken over is rejected.

For instance, in the sample schedule of Figure 4.1, the relief time at 0949 on bus 5 is rejected, because no early duty can take over the bus at this time (the minimum mealbreak being 50 minutes), whereas several duties can take over at 1045 or 1119.

If the early bus cannot be taken over by at least four early duties, it is assumed that the first crew working on the bus can hand over either to a crew starting the first stretch of a day or middle duty, or to a crew which has just worked a peak-only bus and can take over following a joinup instead of a mealbreak. To allow for the first case, any time within the first time-band which is late enough to be the start of a day or middle duty is selected. The program also looks for potential joinups, and for each one checks that a complete valid duty can be formed. The end of the second spell following the joinup will not normally be a marked time, since the spell must be very short. Possible times at which the second spell can end are selected, and the later forward marked times on the bus are recalculated from the latest such time.

For example, on bus 7 in the sample schedule shown in Figure 4.1, the first crew must be relieved by 0950. No other early duty can take over the bus at this time following a mealbreak, but three peak-only buses (2, 8 and 10) finish early enough for the crew working any of these buses to take over bus 7 following a joinup and to carry on working until 1024 or

1100 before taking a mealbreak. The relief times at 1024 and 1100 are selected, and the first forward marked time on bus 7 becomes 1100 instead of 0950. The later forward marked time on this bus is recalculated and becomes 1504 instead of 1354, so that the relief times at 1430 and 1504 become candidates for selection.



At the end of this stage, a set of potential first stretches of early duty have been found, each of which starts by taking an early bus out of the garage and finishes at a time at which the bus can be taken over by another duty. In addition, every peak-only early bus can be covered by the first stretch of an early duty. (Except on the peak-only buses, there will in general be more than one possible first stretch of early duty on each early bus, and no decision has been made as to which is preferred.)

Next, these first stretches of early duty are considered in ascending order of finishing time, and the program looks for possible second stretches for each one.

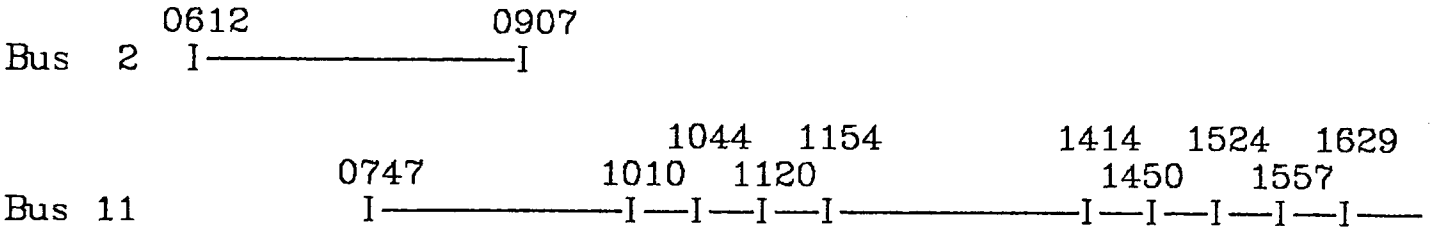
The program attempts to find a possible second stretch on every other early bus for each of the first stretches. However, since the relief times used by the possible second stretches found in this stage may be selected for use by the duty generation program, and the number of selected relief times should be kept as low as possible, the program first attempts to find second stretches using relief times which have already been selected (i.e. relief times used as starting or finishing times by the second stretches assigned to the first stretches already considered; initially, of course, there are none).

If no second stretch can be found on a particular early bus, the search is repeated, and any relief time falling within a time-band is then eligible. A pair of such relief times is found for the starting and finishing times of the second stretch. Provided that the second stretch gives a valid duty in combination with the first stretch, the starting time of the second stretch should be as early as possible (giving the shortest mealbreak) and then the finishing time should be as late as possible.

For each possible first stretch of early duty, only the "best" second stretches are kept, where the best second stretches are defined to be those which start earliest, i.e. those giving the shortest mealbreak. This is in line with the way duties are actually formed by the duty generation program, as described in the next chapter. The ten best stretches are kept (although this number can be varied by the scheduler) and when the best set of second stretches for a given first stretch has been found, the relief times starting and finishing these second stretches are selected. The next first stretch is then considered.

When all possible first stretches have been considered, if there is an early bus such that no possible second stretch has been found for any of the first stretches covering its first piece of work, the search is repeated for all the possible first stretches on this bus, and any relief time in the schedule can now be chosen to form the second stretch.

In the sample schedule of Figure 4.1, the earliest finishing first stretches are on the peak-only buses, and the earliest of these is bus 2, from 0612 to 0907. The second stretches considered for this first stretch must be on early buses which carry on working through the middle of the day. The earliest relief time, chosen from those within the time-bands, at which a second stretch can start is 1010 on bus 11.



Since the maximum stretch length is 4:36, this stretch must finish by 1414, and the duty would then be valid, so the relief times at 1010 and 1414 will be selected.

The latest finishing first stretches (for instance, 0747 to 1154 on bus 11 in the sample schedule) cannot be matched with second stretches in this way. The finishing times of these stretches have been selected because they are suitable starting times for second stretches of duty whose first stretches finish earlier on other buses, and eventually the chaining of early duties from one bus to another must terminate. However, if the relief time at 1154 on bus 11 was chosen as a changeover time in the final schedule, the stretch from 0747 to 1154 could be the first half of a split duty. Alternatively, the first stretch of work on bus 11 could finish earlier, say at 1010 or 1044, and possible second stretches of early duty have been found to go with these first stretches (or again, these first stretches could be the first halves of split duties). So the selection of relief times in the early part of the day gives the choice of many different

ways of covering the work on each bus.

4.6 Late Duties

The treatment of late duties is in some respects similar to that of early duties, working backwards from the end of the day rather than forwards from the beginning, but because of the different patterns of bus operation at the opposite ends of the day, there are some important differences.

First, all possible last stretches of late duty are considered: they must be on buses which finish no earlier than the earliest finishing time for late duties, and they must start at a relief time within the last time-band on the bus.

The possible last stretches are sorted in ascending order of start time, and then the program attempts to find first stretches to go with each last stretch. Each first stretch must be either on another late bus or on a bus returning to the garage after the evening peak, in which case the stretch must finish at the garage. As with the early duties, the relief times used by the possible first stretches assigned to each last stretch will be selected, and the number of selected relief times should be limited, so for each bus on which a possible first stretch may exist, the program first tries to use only those relief times which have already been selected, if any (i.e. relief times used by the first stretches assigned to the last stretches already considered). If no first stretches can be found, other relief times which fall between a pair of marked times can be considered; as before, a first stretch is found which gives a valid duty when combined with the last stretch and such that the mealbreak is as short as possible and the starting time is as early as possible. For each last stretch, only the "best" first stretches are kept and the relief times used by these stretches are selected; as with the early duties, the "best" stretches are those giving the shortest mealbreak, and normally up to ten are kept.

When all possible last stretches of late duty have been dealt with in this way, their starting times are considered. If the starting time has not yet been selected, this indicates that the relief time has not been used as a finishing time for any of the possible first stretches of late duty just constructed. The relief time could, however, be used as the finishing time of a middle duty, provided that it is early enough. If so, the relief time is selected, whereas a relief time at the end of a first stretch of early duty at which the bus cannot be taken over by the second stretch of any other early duty is rejected (unless it appears that a day or middle duty or a

joinup may be required). As explained in section 1.8, the efficient coverage of late buses requires the construction of mealbreak chains, which usually terminate when the end of a late bus is covered by the second stretch of a middle duty handing over to the second stretch of a late duty. The changeover in that case would often be at a relief time which is too late to be the end of the first stretch of a late duty, so that rejecting such relief times would not allow the construction of efficient mealbreak chains. Hence, any relief time within the last time-band on a late bus is selected, provided that it can be either the finishing time of a middle duty or a possible changeover from the first stretch of one late duty to the second stretch of another.

In the sample schedule of Figure 4.1, no first stretch of late duty can be found to finish between 1950 and 2100 on bus 10, or at 2034 on bus 9, but all these relief times are kept. The IMPACS schedule in fact has a middle duty finishing at 1950 on bus 10.

A check is made to see that suitable starting times for the second stretch of any potential middle duty have been selected: extra relief times (within the time-bands) will be chosen if necessary. The same thing is done for the last stretch on any bus if it finishes too early to be the second stretch of a late duty, but too late to be the first stretch of a late duty.

At this point, relief times have been selected for the starting and finishing times of potential second stretches for early duties and first stretches for late duties. On most running boards, at least one relief time from each time-band will have been selected; any omissions are dealt with as described in the following section.

4.7 Tidying Up

Finally, the set of selected relief times is examined, to make sure that there are no long gaps. At least one relief time within each time-band must be selected: if none has so far been chosen, then all of them are. The gap between two adjacent selected relief times must not be greater than the maximum spell length specified. This can only happen if there is at least one relief time between the two which is within a time-band but has not so far been selected; one or more of these times is added to the set of selected times until the gap is short enough.

4.8 Selection Procedure For London Transport

The method for finding forward and backward marked times described above was used for most of the development work on IMPACS.

However, when implementation for London Transport began, one of the clauses of the union agreement raised a difficulty:

> "No scheduled spell of duty shall exceed 5 hours in charge of the vehicle. On Monday to Friday where the scheduled time on duty exceeds 7 hours 38 minutes or on Saturday where the scheduled spreadover exceeds 8 hours 18 minutes the maximum spell in charge shall be 4 hours 30 minutes."

(Time in charge of the vehicle includes signing on or off and other allowances; a spell is the same as a stretch in this context.)

Hence the maximum stretch length has different values depending on the characteristics of the complete duty, which is of course unknown at the time when the forward and backward marked times are being calculated.

In fact, the agreement has since been changed so that the maximum time on duty is now 7 hours 36 minutes, and the clause now only applies to Saturday schedules, but as another clause of the agreement undertakes to minimise the number of stretches which are longer than 4:30, for all schedules, there are still in a sense two different values of the maximum stretch length, an absolute maximum and a desired maximum.

It is important to allow stretches of up to five hours to be formed, but if buses are marked into time-bands using a spell length of five hours less allowances, there may be buses on which stretches of 4:30 or less cannot be formed using relief times within the time-bands. Under the original agreement, this might limit the duties which can be formed, and might in any case lead to a schedule with too many long stretches. Conversely, if a spell length of 4:30 less allowances is used, it may on some buses be impossible to construct stretches of five hours, even though this may be desirable, for instance to allow the first stretch of early-starting duties to cover the whole of the morning peak.

In order to cater for the two different maximum stretch lengths, a new strategy for marking forward and backward marked times was devised.

Suppose that the driving time in a stretch is ten minutes less than the total stretch length (as would normally be the case in a London Transport schedule operated by a driver and conductor), so that the maximum spell lengths are 4:20 and 4:50.

Any running board which is 4:50 or less from start to finish can be covered in a single stretch and will not be broken: the scheduler has to

select relief times manually to alter this.

On longer running boards, the program first finds the latest relief time which is not later than 4:50 from the start of the bus. This becomes the first forward marked time. The program also finds the latest relief time which is not later than 4:20 from the start of the bus, and uses this as the time from which to start counting forward again.

For instance, if a bus leaves the garage at 0530 and passes a relief point every half hour, the first forward marked time will be at 1000. However, the latest relief time which is not more than 4:20 from the start is 0930, and this is the time from which the program counts forward to find the second forward marked time (at 0930 + 4:50 or earlier). The forward marked times on this bus will be at 1000, 1400, 1800 and 2200, supposing that the bus returns to the garage at 2330.

The program next finds backward marked times by working backwards from the end of the bus in a similar way. In the example, the last backward marked time on the bus will be at 2330 - 4:50 or later, (i.e. 1900), but then using the 4:20 stretch length, the program will work backwards from 1930 to find the next marked time.

The time-bands on this bus are 0700-1000, 1100-1400, 1500-1800 and 1900-2200. If the 4:50 spell length alone had been used, only the relief times at 1000, 1430 and 1900 would have been included in the time-bands, and with a 4:20 spell length the time-bands would have been 0730-0930, 1130-1330, 1530-1730 and 1930-2130. Using the two different spell lengths ensures that it is always possible to choose a spell of not more than 4:20 driving time, but at the same time allows more scope to choose longer stretches than if just the 4:20 spell length had been used. For instance, if the bus had left the garage at 0730 instead of 0530, then with a spell length of 4:20 the forward and backward marked times would have coincided at 1130, 1530 and 1930, so that it would not have been possible to choose a longer spell on this bus.

Although the strategy of using two different spell lengths when calculating the marked times was developed to cater for a feature of London Transport schedules, it can also be useful for other schedules, and this has now been adopted as the general strategy. If the two spell lengths, which can be thought of as the maximum spell length and the average spell length, are equal, the results are exactly the same as with the earlier procedure.

4.8.1 Manual Selection Procedure For London Transport

It has been difficult to get good results for London Transport

schedules using the automatic selection procedure. This is partly because the quality of the schedule is very important. A crew schedule which covers all the work in the minimum number of duties and keeps the total cost low may still be unacceptable because there are many different undesirable features to be avoided. These are described in chapter 9. Moreover, many London Transport schedules are relatively small, and because many of the bus routes through central London are very long, the relief times are infrequent (usually more than an hour apart on average). With these schedules, there may be little freedom of choice when constructing the crew schedule, especially when the quality of the duties is important.

If possible, London Transport schedulers prefer to keep all the relief times in the problem, in order to improve the quality of the schedule, even if the program then takes longer and hence is more expensive to run.

L.T. schedulers therefore use the selection procedure only for relatively large schedules where it is necessary to cut down the number of relief times considerably. For other schedules, they have been provided with a facility which allows them to specify the earliest and latest relief times which can be used as changeover times on each bus; in addition, all the intermediate relief times on short running boards which can be covered in one spell of duty are banned.

For small schedules, the relief times which are banned by the scheduler at the beginning and end of each bus are usually those which would not have been used by the duty generation program anyway, but for larger schedules, more relief times are rejected and the scheduler exercises judgment to decide that some relief times which the duty generation program would otherwise use are not in fact necessary.

4.8.2 Results For London Transport Schedules.

Recent experiments have shown that the selection procedure gives much better results for L.T. schedules if the pairs of marked times calculated initially cover a wider range of relief times. This is done by keeping 5 hours (less allowances) as the longer maximum spell length, but reducing the shorter "maximum spell" length to 3:30 (less allowances). The rationale for this is that the average time on duty in London Transport schedules is usually about 7 hours, so that the average stretch length is about 3:30. By using these two values, the buses can be divided into spells which are about the average length of a stretch, but at the same time it is possible to choose stretches up to the maximum

length allowed.

With these values, the selection procedure gives much better results than formerly, although since more relief times are selected, the time saving is reduced. The selection procedure is still inappropriate for some L.T. schedules, however, because of the small size of the schedules and the infrequency of the relief times.

Table 4.1 shows the results of using the selection procedure on some larger schedules, where it is appropriate to use it. In most cases, the problems were small enough for the selection procedure to be not strictly necessary; the lines marked (A) show the results of the selection procedure; those marked (B) show the results of either using the relief times selected by a London Transport scheduler, or allowing all the relief times to be used.

The composition of the objective contribution of each duty is described in chapter 6: for the problems in Table 4.1 it includes a measure of the quality of each duty, as well as the cost in wages. All the work must be covered by the number of duties given in the second column of the table. The cost of the schedule is the total objective contribution of the duties finally selected: it may be less than the cost of the best integer solution found by the ZIP program because of the heuristic improvements described in chapter 8, and also in some cases because any over-cover in the integer solution (which contributes to the cost) must be removed to arrive at a feasible schedule.

For the first three problems, the schedules found by using the automatic selection procedure are compared with those compiled using relief times selected by an L.T. scheduler. By coincidence, the number of relief times selected by the two methods is almost the same. However, the scheduler's selection only bans relief times at the beginning and end of each bus, whereas the automatic procedure rejects times throughout the day. The duty generation process was run in exactly the same way using the two sets of relief times, so that the number of variables is also similar for the two different approaches. The schedules obtained are roughly comparable in quality, and since the set covering problems are very similar in size, the computer times are also approximately the same.

The fourth problem (PM36B) was run in a similar way, except that for schedule (B), no relief times were banned before the duty generation program was run. The reduction from 285 to 251 pieces of work is simply due to the fact that some of the relief times cannot be used to form duties anyway. Using the selection procedure in fact gives slightly better

Problem	No. of duties	No. of pieces of work	Average length of piece (min.)		No. of con-straints	No. of variables	L.P. optimum	Time (sec.)	Best integer found	Time (sec.)	Cost of schedule
AP86	50	311	64	(A)	225	3564	127283.7	68.9	127544	7.3	127501
(Mon-Fri)				(B)	226	3549	126546.0	58.9	126809	19.9	126720
AP86	44	274	63	(A)	212	3163	120484.8	82.0	120623	3.9	120623
(Sat)				(B)	211	3100	120520.7	85.5	120745	10.7	120739
SW77A	59	358	64	(A)	262	3831	150182.1	75.0	150378	24.6	150378
(Mon-Fri)				(B)	256	3686	150270.1	65.8	150426	7.2	150426
PM36B	67	285	95	(A)	226	4325	169608.7	55.2	169782	7.0	169775
(Mon-Fri)				(B)	251	4946	169703.6	83.8	169892	26.0	169849
U147	19	143	51	(A)	97	2350	50987.8	15.5	52324	34.3	52128
(Mon-Fri)				(B)	126	4738	50540.7	40.4	None found		
CF24A	52	458	45	(A)	270	4234	139363.7	66.5	139923	26.3	135613
(Mon-Fri)											
U101	32	274	43	(A)	197	3343	86716.4	57.4	87026	13.5	87008
(Sat)											
TC130	23	251	34	(A)	139	2432	58283.7	16.3	58632	7.9	58613
(Mon-Fri)				(B)	191	2298	58904.8	24.8	59394	9.0	58696

Table 4.1 Effect of Selection Procedure on London Transport Schedules

results than keeping all the relief times, and the solution is found more quickly, as expected.

The U147 problem would not normally be a candidate for the selection procedure, since it has fewer than 150 pieces of work initially and only 19 duties are required. However, the number of split duties in this schedule was restricted to four, and with this limit it proved impossible to find a schedule using the full set of relief times; schedule (B) in Table 4.1 represents one of several attempts. Using the selection procedure was, however, successful in finding an acceptable schedule in this case.

The final three problems shown in Table 4.1 are cases where the selection procedure was necessary or advisable in order to reduce the size to a manageable level. The CF24A problem had too many pieces of work, and rejecting relief times at the beginning and end of each bus would not have been sufficient in this case. The IMPACS schedule was produced for comparison with an existing manual schedule, and was of similar quality.

The U101 problem shows that it may sometimes be necessary to use the relief time selection procedure not for its primary purpose of reducing the number of constraints, but in order to reduce the number of duties that can be generated. Although there were originally 274 pieces of work, which would be small enough to cope with, the duty generation program formed over 21,000 duties, even when the duties generated were restricted as much as was considered safe. The reduction programs described in section 5.11 could not in this case reduce the set of duties to the size required by ZIP. The difficulty arises because of the frequency and regularity of the relief times, giving an enormous number of possible combinations of spells, so that again it is essential to ban relief times during the middle of the day, not just at the beginning and end of each bus. The selection procedure was used, cutting the number of pieces of work to 197. The generation program then formed only 10,000 duties, and a 32-duty schedule was found without difficulty.

The last example (TC130) is similar to U101 in that although the number of pieces of work is not too large, the duty generation program formed more than 19,000 duties. In this case, the reduction programs could be used (because the COMPARE program described in section 5.11.2, which deletes any duty which is wholly included in another duty, was able to reject most of the 19,000 duties), and eventually the number of duties was sufficiently reduced to allow the set covering problem to be solved. However, the result was not as good as that found using the selected

relief times only. By reducing the number of pieces of work from effectively 191 to 139, the number of duties which can be formed by the duty generation program is reduced from 19,000 to just over 4,000. Table 4.1 shows that the ZIP program took longer to run for the larger problem, and also generating 19,000 duties and then rejecting most of them took much longer than forming the set of duties using the selected relief times. Although the full set of relief times gave an acceptable schedule, the selection procedure gave a better schedule in much less computer time.

These examples demonstrate that the automatic selection procedure now works well for the larger London Transport schedules. It should be noted that the examples include both weekend and weekday schedules; the method is not affected by the presence or absence of morning and evening peaks.

4.9 General Results

As mentioned earlier, it was recognised from the start that the selection procedure should be tested in a wide variety of circumstances to make sure that it did not depend for success on a particular form of union agreement or pattern of bus operation.

Table 4.2 illustrates the different schedules to which the selection procedure has been applied. The schedules are described in detail in chapter 9. Except for the Leeds schedules, where only a TRACS schedule was available for comparison, and the Greater Manchester (Bolton) schedule where for reasons explained in chapter 9 the IMPACS schedule required an extra piece of overtime, the number of duties in the IMPACS schedule is always equal to or less than the number of duties in the manual schedule. The number of duties is shown in the second column of the table, in some cases consisting of a number of full duties and a number of pieces of overtime (marked o.t.). Where the complete manual schedule is available for comparison, the individual duties in the IMPACS schedule are of similar quality to those in the manual schedule, as far as can be discerned, and whenever the IMPACS schedule has been discussed with schedules staff of the companies concerned, they have agreed that the schedules shown in Table 4.2 are acceptable. This demonstrates that good schedules can be found using the selection procedure.

There were considerable differences between the union agreements applicable to the schedules in Table 4.2. There were differences between bus companies, and also in some cases between different garages operated by the same bus company. The schedules were for weekdays,

Problem	No. of duties in IMPACS schedule	No. of pieces of work	Average length of piece (min.)	No. of constr-aints	No. of variables
Leeds schedules:					
A	19	263	31	67	697
B	44	465	41	158	1794
C	31	230	49	75	535
D	72	695	44	234	2817
Leicester	59	651	44	226	2415
Cleveland Transit North Depot:					
Mon-Fri OMO	88 + 8o.t.	578	62	296	3530
Mon-Fri TMO	48 + 7o.t.	418	48	197	2405
Saturday TMO	44	394	43	192	1810
Sunday TMO	24	182	47	93	993
South Depot:					
Mon-Fri	38 + 3o.t.	362	38	135	2011
Saturday	31	342	32	111	1888
Sunday	22 + 2o.t.	237	34	103	1213
West Midlands PTE Selly Oak garage:					
Mon-Fri	75	514	60	223	2482
Saturday	64	547	48	233	2676
Yardley Wood	33 + 2o.t.	182	75	91	1835
Cardiff	81 + 10o.t.	627	52	231	2862
East Kent Canterbury	58 + 12o.t.	330	67	181	2791
Yorkshire Traction Wombwell	72	351	79	221	3743
Rawmarsh	75	410	70	248	3663
Greater Manchester Bolton	36 + 2o.t.	299	49	159	1767
Melbourne	41	446	39	164	2213
Tyne and Wear South Shields	140	1119	44		
A *	72	563	45	288	3132
B *	82	667	44	328	4015

Table 4.2 Problem Size Following Selection

* As explained in chapter 9, the South Shields schedule was compiled as two separate subproblems.

except where otherwise stated, with morning and evening peaks in the number of buses and crews required, but whereas in some cases the extra peak requirement was covered almost entirely by split duties (e.g. Leicester), in others, no split duties were allowed (e.g. Cleveland), or only a limited number (e.g. Cardiff). The weekend schedules had no peaks and usually no split duties were allowed. The selection procedure is sufficiently adaptable to produce good results in all these cases.

As can be seen, although the number of pieces of work in these schedules ranges up to nearly 700, set covering problems of manageable size can be obtained for all of them by reducing the number of relief times which can be used. Without the selection procedure, it would be impossible to solve most of these problems.

4.10 Sample Results

Figure 4.3 shows the relief times selected from the bus schedule shown in Figure 4.1. Only a small proportion of the relief times have been selected: the original 203 pieces of work have been reduced to only 65 constraints in the set covering problem. (An IMPACS schedule based on this set of selected relief times is shown in Figure 8.1)

Because a set covering problem with 203 constraints could theoretically be solved, it might be thought that the selection procedure is not necessary in this case, particularly as the duty generation process will itself reduce the number of relief times actually used. However, whereas the 65 pieces of work left by the reduction gave rise to 1367 generated duties, when no relief times were banned, more than 48,000 duties were generated before the attempt was abandoned.

By restricting the generation process considerably, for instance by reducing the maximum mealbreak allowed in a duty and increasing the minimum length of a stretch, the number of duties generated using all the relief times was reduced to about 22,000, and by applying the reduction methods of section 5.11, a set covering problem with 2747 variables and 182 constraints was eventually formulated. However, the branch-and-bound algorithm failed to find an integer solution before the search terminated after 50 nodes had been developed.

By contrast, there was no difficulty in finding an integer solution for the much smaller set covering problem based on the selected relief times; in fact, the solution to the relaxed L.P. was itself integral.

To confirm that the difficulties in the larger problem were not due to the extra restrictions placed on the duty generation process, a final set covering problem was set up, using the selection procedure and

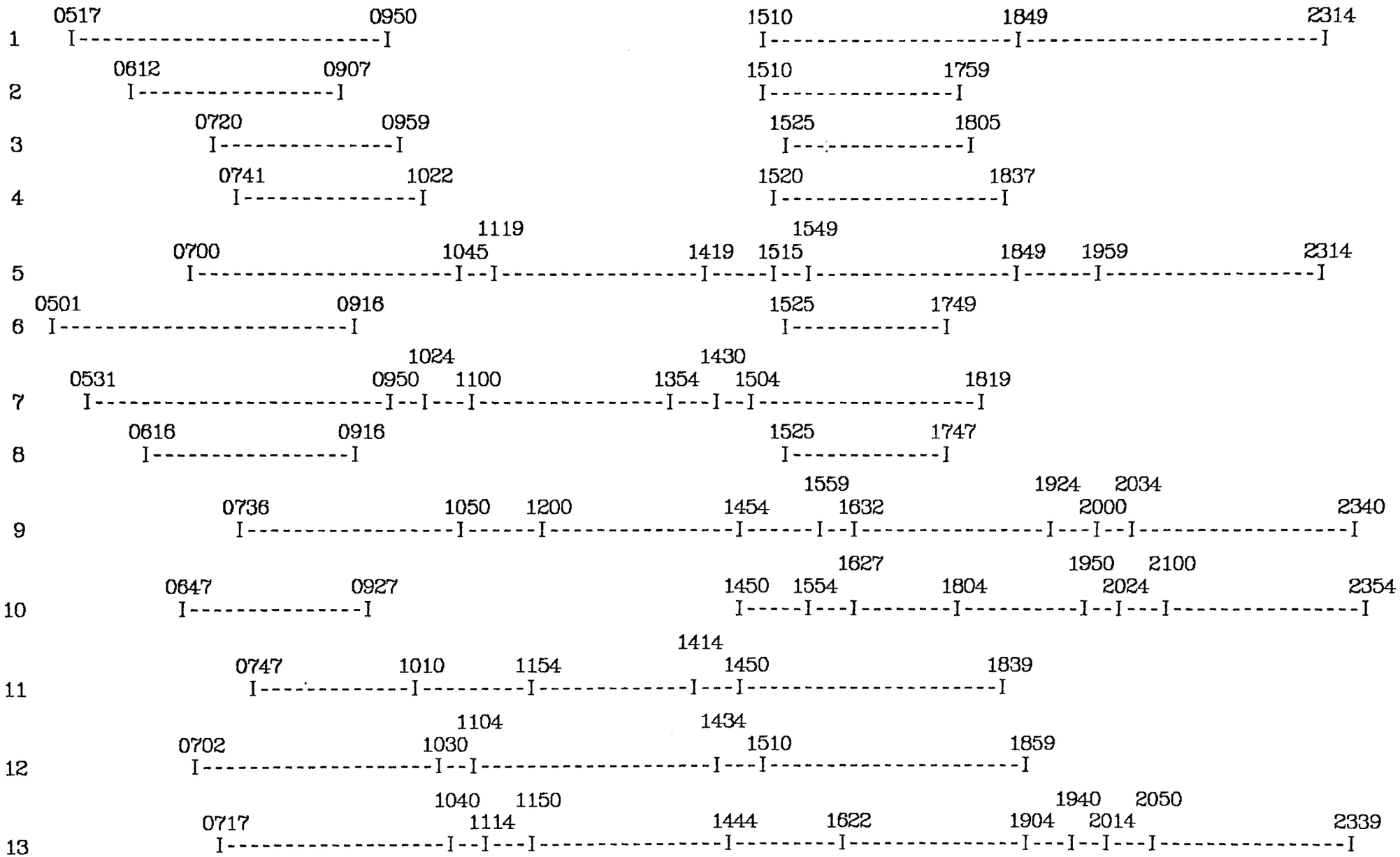


Figure 4.3 Selected Relief Times For A Sample Schedule

restricting the duties generated at the same time. The solution to the relaxed L.P. was again integral. (This solution is much more expensive than the previous one, because there is an over-covered piece of work in the schedule, contributing 1330 to the total cost. The occurrence of over-cover is quite common when the duties generated have been restricted so that only the most individually-efficient duties can be formed.) The results are summarised below.

	No. of constraints	No. of duties formed	No. of variables	L.P. optimum
Normal duty generation:				
Selected relief times	65	1367	1367	48906 (integer)
No selection	<203	>48000		
Restricted duty generation:				
Selected relief times	68	873	873	50365 (integer)
No selection	182	22156	2747	48742.9 (no integer solution found)

The experience with this schedule is an extreme example of an additional benefit of reducing the number of constraints: integer solutions are easier to find, in the sense that the branch-and-bound algorithm needs to do less development of the tree. This is distinct from the reduction in branch-and-bound time resulting directly from the fact that reducing the number of constraints reduces the time taken to solve each subproblem in the tree. Although the size of the tree formed by the branch-and-bound algorithm is very variable, so that the effect is hard to quantify, experience shows that it is more difficult to find integer solutions if the number of constraints is large in relation to the number of duties required in the schedule. Conversely, the relief time selection procedure, by reducing the number of constraints, makes integer solutions easier to find.

CHAPTER 5. DUTY GENERATION

5.1 The Duty Generation Program

The duty generation program forms the initial set of valid duties from which, possibly after some reduction as described in section 5.11, the schedule will be selected.

Any duty formed must pass a number of validation checks. These are partly controlled by a set of parameters, specified by the scheduler in a duty generation parameter file, which is described in detail in the Appendix. The values of many of the items in this file will be determined by the union agreement, and to a considerable extent this file is used to incorporate the agreement into IMPACS. The validation of potential duties is completed by a call to a routine, VALID, which caters for any rules in the agreement which are not covered by the parameters; there is a separate version of VALID for each agreement. Similarly, for each agreement there is a WAGES routine which calculates the cost of each duty, measured usually in terms of paid time; both of these routines will be described more fully in chapter 9.

Not all legally valid duties are formed; there are some limitations built into the program, which are described below, and after the relief time selection program has been run, only duties using selected relief times can be formed. Further, although many of the parameters which are referred to when any potential duty is validated reflect clauses of the scheduling agreement, others are designed to allow the scheduler to eliminate duties which are unlikely to be used in the schedule, although legally valid. Generally, these are duties which contain much less work than the average, and the scheduler would eliminate them by specifying appropriate values for minimum cost, maximum mealbreak and so on.

Hence, valid duties, as far as the duty generation program is concerned, must be both legally valid, as defined by the agreement, and sensible, as defined by the scheduler.

5.2 First Available Pieces

The pieces of work are first sorted in ascending order of starting time, so that whenever a duty transfers from one bus to another, it is easy to find the first piece of work which the duty can do next, following a mealbreak or a joinup. Since considering all possible next pieces would create an enormous number of duties, only a subset of next pieces can be considered and those which start earliest are a logical choice.

Except for split duties, which must necessarily have a long

mealbreak, duties with short mealbreaks are usually preferable to those with long breaks (especially if the mealbreaks are paid). A preference for duties with short mealbreaks may also have some influence on the integrality of the L.P. solution. Ryan and Foster [26] argue that if duties are constructed so that the driver always does the first available piece of work, the L.P. must have an integer solution (in fact, all basic feasible solutions are integer). Unfortunately, since in most cases the first available piece of work is the immediately subsequent piece on the same bus, the requirement for mealbreaks off the bus destroys the pure first available structure. Ryan & Foster claim that if duties are constructed so that when a mealbreak is required, the next piece of work done is the first available following the completion of the mealbreak, the L.P. is still likely to have an integer optimum, although natural integrality cannot be proved.

However, unless duties are also constrained to stay on the first bus for as long as possible before taking a mealbreak, there are usually several possible relief times at which the mealbreak can start. Even if only the first available second stretch is associated with each mealbreak start, a considerable number of possible duties can be formed to cover each piece of work, and the L.P. optimum is often non-integer.

Moreover, it is not possible to form good schedules using only this type of duty; apart from the complications of split duties, which must have a long break, the duties formed give only a very restricted choice, so that although it may be easy to find solutions, the resulting schedules are unacceptable. It has been found necessary to extend greatly the number of next availables considered when forming duties; usually up to ten are considered.

Although the set of duties formed is very far from having the ideal first available structure, where possible the ideal has been borne in mind, so that when it is necessary to choose between two duties, the one which has the longer mealbreak is rejected and the one which is nearer to using the first available piece is kept.

5.3 Types Of Duty

Five types of duty are formed:

a) Early duties, taking buses out of the garage before the morning peak;

b) Day duties, which either take over early buses on the road from another crew, or occasionally take a later-starting bus out of the garage;

c) Late duties, working on late evening buses returning to the garage;

d) Middle duties, which either finish at the garage shortly after the evening peak, or hand over to late duties on the road;

e) Split duties (if required).

The characteristics of the different types of duty are controlled by the values of various parameters which can be specified by the scheduler, e.g. earliest sign-on time, maximum spreadover and so on.

5.4 Two-bus Duties

Two-bus duties are formed as two spells of duty separated by a mealbreak. The formation of early duties will be described in detail, using an example.

Figure 5.1 shows the bus graph of a small sample schedule. Each bus starts and finishes at the garage G, and passes a relief point R outside the garage at the times shown on the graph. It will be assumed that the relief time selection program has not been run, so that any of the relief times can be used to change crews, provided that valid duties can be formed.

The duty generation program begins by forming early duties. According to the definition used during the duty generation process, early duties must start at the beginning of a running board. The buses are considered in the order in which they appear in the bus schedule. In this case, the first bus in the schedule which leaves the garage early enough to be covered by an early duty is bus 161, and a set of early duties starting at 161:0621 will be formed. For the first of these duties, the end of the first stretch is taken as the earliest time which gives a stretch exceeding the minimum length defined by the scheduler for the first stretch of an early duty, and which is later than the earliest starting time allowed for a mealbreak. The minimum length of a stretch is not usually defined in the agreement, but is an extra parameter designed to reduce the number of duties formed. For the very early starting duties, however, the end of the first stretch is likely to be governed by the canteen opening time. In the example, the first stretch considered on bus 161 ends at 0805.

After the first stretch, there is a mealbreak, and the duty resumes by picking up the earliest piece of work which will give a legal mealbreak. Assuming a minimum mealbreak of 40 minutes, this is the piece starting at 0907 on bus 164 in the example. (Although two-bus duties normally work on two different buses, it is possible for the driver to return to the same bus, provided that the intervening piece of work, when the driver is taking a mealbreak, is long enough to form a reasonable spell of another duty.) A potential duty is formed ending at the earliest relief time which

```
          0702                                                                                  2028       2228
        0621    0805  0915 1017  1127  1237  1347  1457  1607 1717  1828  1932       2132          0003
161      G--R----R-----R----R-----R-----R-----R-----R-----R-----R-----R-----R---R-----R---R-------G


                      0935                                                          1925
    0541       0731 0835       1037  1147  1257  1407 1517   1630 1739  1843
162  G---------R----R----R----R-----R-----R-----R----R------R----R-----R--G


                0815    0948  1057  1210  1319 1427  1537  1647  1757 1858 2002 2058 2202 2258 0005
163              G-------R-----R-----R-----R----R-----R-----R-----R----R----R----R----R----R----G


                                                                                  2128    2303
          0700 0801  0907 1007 1117  1227  1337  1447  1557  1707 1817  1928  2032       2232
164        G----R-----R----R----R-----R-----R-----R-----R-----R----R-----R-----R---R-----R-G
```

Figure 5.1 Sample Bus Schedule

will give a second stretch at least as long as the minimum length for the second stretch of an early duty.

The duty is then validated: it is checked against any parameters for this duty type which cannot be checked earlier, e.g. maximum spreadover, and the VALID routine is called to check any rules which are not covered by the parameters. The cost of the duty is calculated by the WAGES routine and checked against the maximum and minimum cost for this duty-type.

If the duty is not valid, then either the second stretch is extended to the next relief time, or other second stretches with later starting times are considered, depending on which rule was violated.

If the duty is valid, the action to be taken depends on the scheduling agreement. Many agreements specify a minimum paid time of say 8 hours for a day's work; any duty which would otherwise cost less than this is made up to the minimum. Under these conditions, if a duty costs less than the minimum paid day, and the second stretch of the duty could be extended to the next relief time without becoming invalid or exceeding the minimum paid day, the shorter duty is not worth adding to the list of potential duties, because the longer duty covers the same work and would cost the same. Hence, when a valid duty is found which is shorter than the minimum paid day, it is only accepted if no such longer duty exists. If there is no minimum paid day, or the cost of the duty is already more than the minimum, it is immediately added to the list of duties.

After a valid duty has been found and added to the list, later finishing times on the second bus are also considered, until no further valid duties can be found, for instance because the maximum spreadover or platform time or stretch length has been reached, or because the duty is already working up to the end of the bus.

In the example, the duty finishing at 1227 on bus 164 would give a total driving time of only 5:04. The total cost, in hours paid, would be about 5:30, including signing on and off allowances, if the mealbreak is unpaid. Although this would be legally valid, the scheduler would normally want to avoid forming duties as short as this and would specify a minimum acceptable cost of say 6 hours. In that case, the first duty formed would be:

161:0621-0805 164:0907-1337

This duty cannot be extended to the next relief time at 1447 on bus 164, because a stretch of 5:40 driving time would be too long.

When the second stretch has been extended as far as possible, or if

no valid duties have been found, later starting times for the second stretch, on other buses, are considered in chronological order. In the example, the earliest time after 164:0907 is 161:0915, and again a sequence of possible finishing times is considered, with the following duty being added to the list:

161:0621-0805 161:0915-1347

The process of considering second stretches of work with different starting times does not continue indefinitely. It will terminate if the next piece of work to be considered gives a mealbreak longer than the maximum specified, or if the maximum number of starting times has been reached: for each first stretch of duty, a maximum of ten different starting times are allowed for the second stretch. (The maximum is controlled by a parameter and can be varied if necessary.)

When all possible second stretches have been found, the first stretch is extended to the next relief time if possible, giving in the example 161:0621-0915 as the first stretch, and a new set of second stretches is found.

Assuming a maximum spell length of 4:35, a maximum cost of 8 hours and a maximum spreadover for early duties of 9 hours, with signing on and off allowances totalling 20 minutes, the complete set of early duties which can be formed starting at 0621 on bus 161 in the sample schedule is listed below:

161:0621-0805 164:0907-1337
161:0621-0805 161:0915-1347
161:0621-0805 162:0935-1407
161:0621-0915 164:1007-1337
161:0621-0915 161:1017-1347
161:0621-0915 162:1037-1407
161:0621-0915 163:1057-1427
161:0621-1017 164:1117-1337
161:0621-1017 161:1127-1347
161:0621-1017 162:1147-1407
161:0621-1017 163:1210-1427

Any relaxation in the rules might allow more duties to be formed. For instance, if spells of 4:40 were allowed, duties such as:

161:0621-0915 164:1007-1447
161:0621-0915 161:1017-1457

would have to be added to the above list.

This illustrates the large number of possible early duties that are

formed even for a very small schedule, in spite of the fact that, because only duties longer than 6 hours have been allowed, many legally valid duties have been excluded.

When all possible early duties have been formed, late, split, day and middle duties are formed, in that order. These types of duty are formed in the same way as early duties, except that they can start at any time between the earliest and latest times specified as parameters, and not just at the start of a running board. However, day and middle duties must start at a relief time which has already been used as a finishing time for a spell of work in a duty already formed, unless they start at the beginning of a running board. Except in the latter case, day duties take over from early or split duties, and middle duties take over from early, split or day duties, and there is no point in forming a day or middle duty if there is no possible duty for it to connect with.

5.5 Formation of Three-bus Duties

Most scheduling agreements allow a duty to work on three buses (or more). In a three-bus duty, at least one of the breaks must be long enough for a mealbreak. The other may be a joinup, that is, a much shorter break allowing only enough time to transfer from one bus to another. The two spells of duty either side of the joinup are treated as a single stretch of work, and the joinup is counted as working time. However, some agreements allow for duties to have two mealbreaks, in which case, the restriction on stretch length is applied only to the individual spells of duty, and not to two spells combined. Because of the difficulty of fitting two mealbreaks, as well as a reasonable amount of driving time, into a limited spreadover, duties with two mealbreaks are usually split duties.

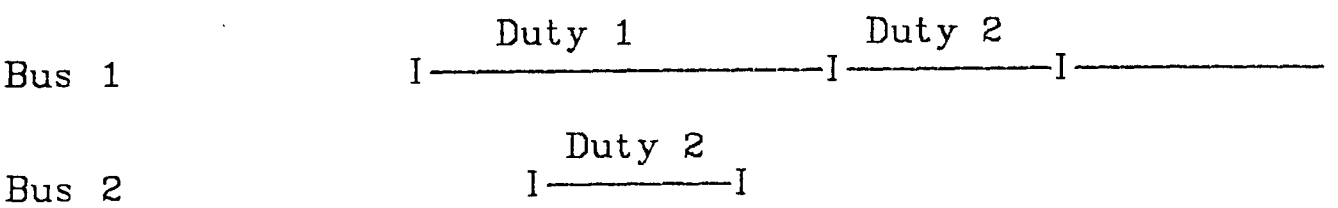
It is often necessary to use three-bus duties in order to achieve a good schedule, but their number should be minimised. Since the joinup is treated as working time, a three-bus duty costs more than a two-bus duty with the same driving time. It may also cause operational difficulties since a joinup gives less opportunity to make up for late running than does a mealbreak.

At the same time, the total number of possible three-bus duties is much greater than the number of possible two-bus duties, and it is very difficult to predict what kind of three-bus duties will be needed. Often far more three-bus duties than two-bus duties have to be generated, even though only a few will be selected for the schedule.

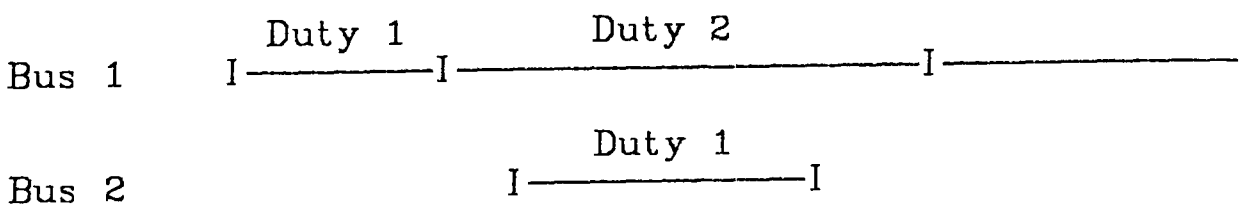
Initially, the generation of three-bus duties was very restricted, and

they were only formed in certain situations. For instance, the most efficient way of covering the morning peak is to assign each morning peak bus to exactly one crew. If the bus returns to the garage after the peak, the crew take their mealbreak then. If the bus carries on through the inter-peak period, the crew continue their first stretch of work until the bus can be handed over to another crew just finishing a mealbreak. However, suppose that the first duty working on an early bus has to be relieved before any other morning peak crew can finish a mealbreak. One solution would be to start another early duty on the bus during the morning peak, but this requires two crews to cover one morning peak bus. It is sometimes possible to solve the problem by using a three-bus duty:

a) If the first duty working an early bus can cover the whole of the morning peak, but must be relieved too soon after the end of the morning peak to hand over to another crew finishing a mealbreak, there may be sufficient time for the crew of a short morning peak bus to take over following a joinup:



b) Alternatively, it may be possible for the first crew to leave the bus before the morning peak, handing over to the second crew then, and also to work a short morning peak bus:



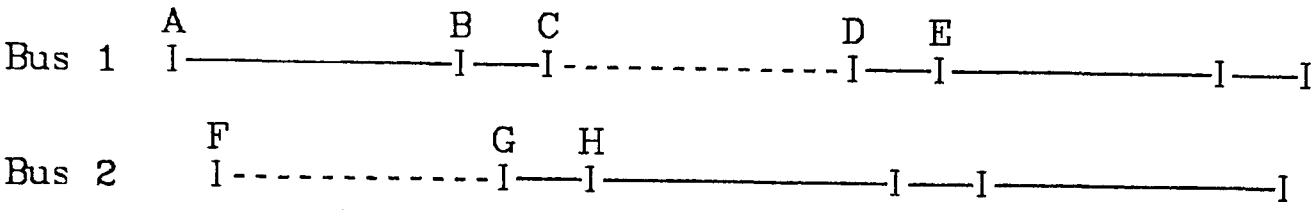
In both of these cases, by using a three-bus duty, in which the second spell is followed by a mealbreak and a further spell, two crews can cover two morning peak buses, thus avoiding the necessity for an extra day duty.

It may also be useful to link a short peak-only running board to another spell of work to give a reasonable stretch length and have a reasonable driving time for the whole duty. If the maximum stretch length is 4:30, say, a two-bus duty including a running board only one hour long could have a maximum driving time of only 5:30.

5.5.1 Three-bus Duties To Extend Stretch Length

As more schedules were compiled, it became necessary to extend the formation of three-bus duties in order to produce satisfactory schedules. A more general reason for using three-bus duties is to allow a stretch to cover more work by changing buses than would be possible by staying on the same bus.

For instance, if the buses work on long routes with the garage (and therefore the relief point) nearer to one terminus than the other, as is common in London, the running boards will consist of alternate "long ends" and "short ends":



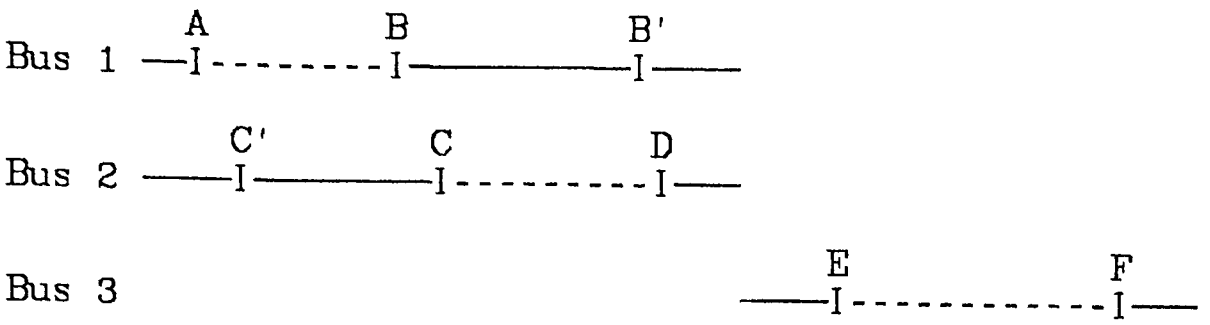
It may happen that two long ends and a short end (e.g. A to D on bus 1) would be too long to be a valid stretch, so that if the schedule were restricted to two-bus duties, no stretch longer than a long end plus two short ends could be formed (e.g. AC and BE would be valid). However, if three-bus duties can be formed, two long ends on different buses, such as FG and CD, could be linked by a joinup, which would allow more work to be covered by individual duties and possibly require fewer duties in total.

The rule adopted to cater for this kind of situation was that a two-bus stretch should be allowed if it covers at least as much work as either of the longest spells which can be formed with the same starting or finishing time. In the example, the two-bus stretch FG, CD would be compared with FH and BD, and since it is in fact longer than both, it would be accepted.

5.5.2 Three-bus Duties To Extend Duty Length

The rule just described proved to be satisfactory for most schedules. However, London Transport schedulers pointed out a few cases of three-bus duties which they felt were necessary for a good schedule but which did not follow this rule. They suggested that in these cases it was not sufficient to consider the stretch of duty in which the joinup falls; the whole duty must be considered.

Suppose the joinup is after the first spell, and the duty is composed of the three spells AB, CD and EF, as shown.



B' is the latest relief time on Bus 1 such that AB', EF is a valid duty. Similarly, C' is the earliest time on Bus 2 such that C'D, EF is a valid duty. The proposed rule was that the three-bus duty should be allowed provided that it covers at least as much work as either of these two-bus duties (AB', EF and C'D, EF).

This would allow more duties than the previous rule. For instance, if AB'' is the longest valid stretch starting at A and C''D is the longest valid stretch finishing at D, then if both of these cover more work than the two-bus stretch AB, CD no three-bus duty containing this stretch would have been allowed. However, when the third spell EF is considered, it may be that neither of AB'', EF and C''D, EF are valid duties (for instance if B'' to E is too short for a mealbreak and C'' to F exceeds the maximum spreadover), so that under the proposed new rule the duty AB, CD, EF would be allowed.

The proposed rule would generate a very large number of three-bus duties, and further consideration suggested that the three-bus duty is more useful if it allows a longer spreadover than if it allows a shorter mealbreak; in other words, the comparison with the duty C'D, EF is more important than the comparison with AB', EF. A study of manually-compiled schedules showed that every three-bus duty used was longer than the two-bus duty which would be equivalent to C'D, EF, with the exception of a few cases in which Bus 1 returned to the garage at B; in these cases, B and B' are the same, and a duty consisting only of AB and EF is clearly inefficient.

The revised rule is therefore that the three-bus duty AB, CD, EF is allowed if either it covers at least as much work as C'D, EF, or Bus 1 returns to the garage at B.

When the joinup is after the second spell of duty, the rule is applied in reverse: the three-bus duty is allowed either if it covers at least as much work as the two-bus duty AB, CD' where D' is the latest time on Bus 2 such that AB, CD' is a valid duty, or if Bus 3 leaves the garage at E.

One of the disadvantages of this scheme compared with the earlier rule is that the complete duty has to be formed before it can be decided

whether it should be kept or not, so that generating three-bus duties takes much longer than formerly.

Far more three-bus duties can now be formed than under the earlier rules. To compensate in part for the increased number of three-bus duties, some further restrictions have been introduced by way of extra parameters. The scheduler can specify that joinups are not allowed at certain times of the day (after the evening peak, for example), and the minimum cost for duties of each type can be specified separately for two-bus and three-bus duties. It is often possible to specify a higher minimum cost for three-bus duties, since they should be relatively long.

Apart from the comparisons with two-bus duties just described and the extra parameters, three-bus duties are formed in a similar way to two-bus duties, with the obvious extensions.

When three-bus duties which have two mealbreaks, instead of one mealbreak and a joinup, are allowed, these duties are formed at the same time as ordinary three-bus duties. A valid duty of this type is worth keeping if it would not be valid when considered as a two-stretch duty with an exceptionally long joinup, because the duty is then sufficiently different from other duties to be potentially useful. If, however, either the first and second spells together or the second and third spells together are short enough to be a valid stretch, the duty must be compared with similar two-bus duties, as described above, before being accepted.

5.6 Formation Of Four-bus Duties

Although many scheduling agreements do not specify the maximum number of buses which a duty can work on, manual schedules with duties working on more than three buses are unusual. The duty generation program does not form duties working on more than three buses, and so far this has not caused any problems in compiling satisfactory schedules. The number of possible four-bus duties is much greater than the number of possible three-bus duties, so that if four-bus duties were necessary for any schedule, it would only be practicable to generate a very small proportion of the total, and the four-bus duties required would have to be carefully defined.

5.7 One-bus Duties

Duties working on only one bus may be either overtime pieces or full duties without a mealbreak, or represent half a duty of which the other half would be made up with a period of stand-by or report time. Many schedules do not allow any of these. In cases where they are

required, it is necessary at present to write a special-purpose routine, called by the duty generation program; this is easily done because of the simple structure of one-bus duties.

5.8 Split Duties

Although defined in many different ways, split duties are basically duties with a longer spreadover than other types of duty: the maximum may be 12 hours or more. Normally, however, the maximum driving time is the same as for other duties, so that split duties have a long mealbreak.

When forming split duties, this characteristic requires that different rules should be followed than for straight duties. For other types of duty, it is reasonable to favour duties with short mealbreaks and to avoid forming duties with long mealbreaks. However, split duties should have long mealbreaks and in the formation of split duties it is important to ensure that duties with long breaks can be formed.

For two-bus split duties, this is done by following the same basic process as for straight duties, but with several different ranges of mealbreak lengths. For instance, if split duties can have mealbreaks between 2 hours and 5 hours long, a group is first formed with a mealbreak between 2 and 3 hours long, considering up to ten possible next pieces for each first stretch, just as for straight duties. Then the process is repeated to form split duties with a mealbreak between 3 and 4 hours long, and finally splits with mealbreaks longer than 4 hours are formed.

This process often generates a very large number of two-bus split duties and would be impracticable for three-bus split duties. Instead, there is no attempt to cover the whole range of mealbreak lengths, since three-bus split duties should have relatively short mealbreaks in any case. Duties are formed in a similar way to three-bus straight duties, except that in order to provide a sufficiently wide selection of duties to choose from, they are formed as two separate "sub-types", one having a mealbreak and then a joinup, and the other a joinup followed by a mealbreak. (When duties with two mealbreaks are allowed, these are formed as a third sub-type.)

Forming a very large number of split duties seems to be unavoidable in most cases. Although it is possible to produce descriptions of split duties which cover most cases, there are no firm rules. For instance, the basic purpose of split duties is to cover both peak periods in one duty, and in many bus companies most split duties do

cover both peaks, but it is not usually possible to say that all split duties must cover both peaks, or even at least one peak, and so limit the number of duties formed. As with three-bus duties, because a few duties of this type will be required in the final schedule and it is impossible to predict the kind of duty which will be required, many possible duties must be formed. If the scheduler does have information which allows the split duties to be restricted in a particular case, this can be specified using the parameter values which control the formation of the split duties, or in the VALID routine.

5.9 Prespecified Duties

The scheduler can specify that a particular duty or duties must appear in the schedule. There are many reasons why this might be necessary: for instance, it may have been agreed that the earliest starting duty should be a short one-bus duty, or the garage union representative may work a special duty which should be assigned in advance. The prespecified duties are accepted by the duty generation program as given, and they are not validated. The work covered is in effect temporarily removed from the bus schedule so that no other duties covering the same work can be formed, and the prespecified duties are simply carried through the whole system and printed in the final schedule.

5.10 Running Time Of Duty Generation Program

Although the duty generation program almost always takes very much less time to run than the ZIP program, it can take a significant amount of computer time: in extreme cases, one minute or more cpu time (on an Amdahl 5860).

The time required obviously depends largely on the number of valid combinations of pieces of work into spells of duty, and of spells of duty into duties, which can be formed. In turn, this depends on the number of relief opportunities available to the program, and the lengths of the intervals between them, and also on the duty generation parameter values and the VALID routine for a particular bus schedule.

However, the running time depends also on how soon invalid duties can be rejected. For bus schedules of a similar size, the program takes longest to run when the main determinant of validity is the minimum cost parameter, because then invalid duties have to be completely formed before being rejected. Conversely, when the scheduling agreement contains rules relating to starting time of duties, length of first spell and so on, which can be checked at an early stage in the

formation process, relatively few invalid duties need be competely formed, and the duty generation program takes much less time.

5.11 Reduction Of Set Of Duties After Formation

After the initial set of duties has been formed, the techniques described in this section can be used. They are designed to weed out many of the duties, which will reduce the number of variables in the set covering problem, and so reduce the time taken to solve it.

The reduction techniques can consider the set of duties as a whole, and reject a duty if the existence of other duties in the set makes it not worth keeping, by some criterion. The duty generation program itself does not pay much attention to the duties which already exist when a duty is formed, and cannot in any case consider the duties which have not yet been generated.

5.11.1 Unbalanced Relief Times

When the set of duties formed by the duty generation program is examined, it may happen that a relief time is found (not at the start of a bus) which is used by some duties in the set as the start of a spell of work, but is not used by any duty as the end of a spell of work.

If the variable corresponding to a duty with a spell of work starting at such a relief time has a non-zero value in an L.P. solution, then the piece of work following the relief time must be over-covered, since no duties end a spell of work at that relief time and hence any duty covering the piece of work before the relief time must also cover the following piece. In fact, the L.P. constraint corresponding to the piece of work following the relief time is redundant.

Conversely, if a relief time is used as the end of a spell of work by one or more duties, but never as the start of a spell of work, the L.P. constraint corresponding to the preceding piece of work is redundant.

In order to avoid complicating the formulation, redundant constraints of this kind are not explicitly recognised. Originally, however, an attempt was made to eliminate them by identifying the relief times concerned and deleting the duties using them if the pieces of work covered by each duty were also covered by a sufficient number of other duties. (In order to delete as many of the duties as possible, ten was considered a sufficient number.) If all the duties using one of these relief times could be deleted in this way, the redundant constraint would automatically disappear from the formulation.

Since the number of duties concerned is likely to be small, and

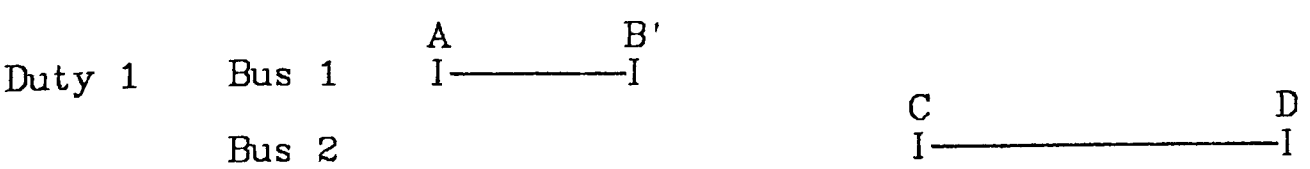
hence the reduction in the number of variables is not likely to be significant, the main benefit of this procedure is in the potential reduction in the number of constraints. However, a constraint can only be deleted if every duty using a relief time can be deleted, and there are usually very few unbalanced relief times, if any, so that the the reduction in problem size is usually negligible.

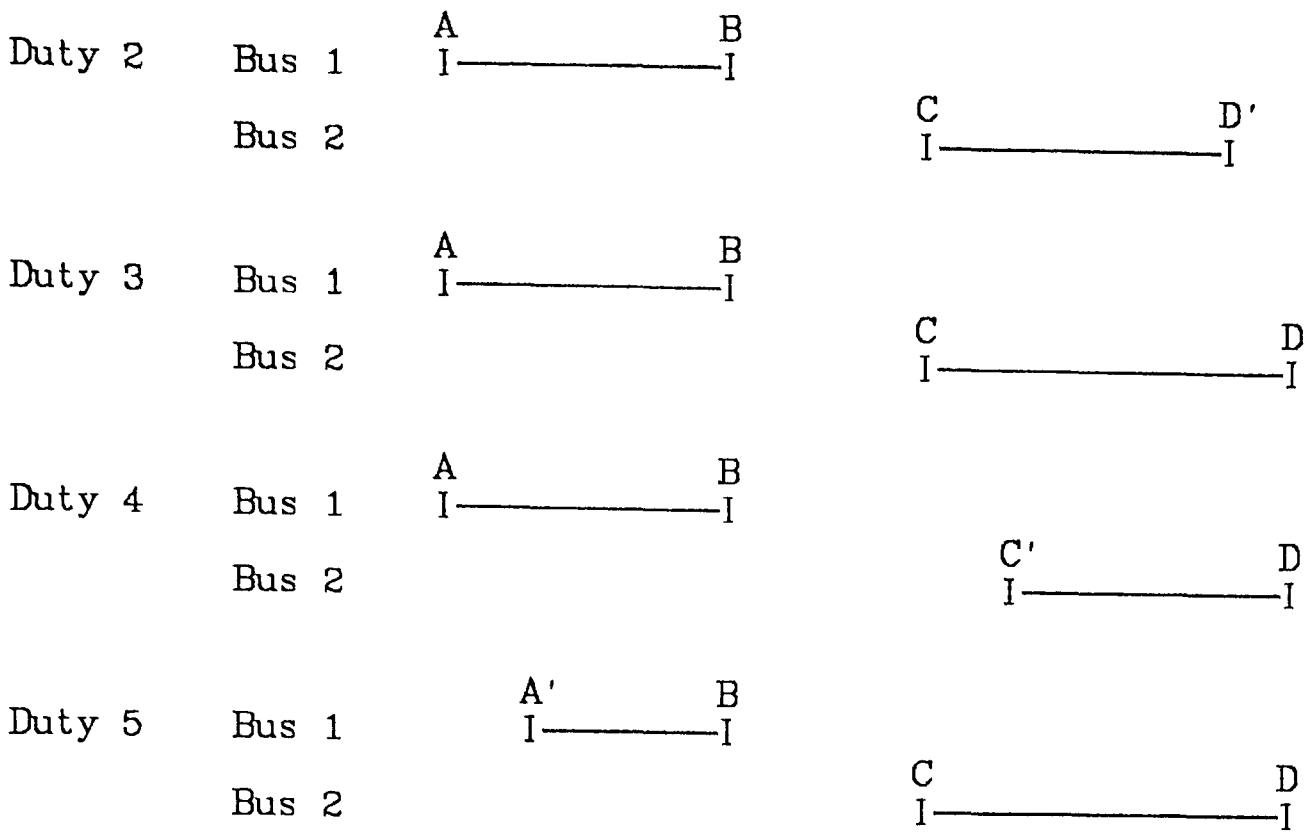
The use of this procedure was eventually dropped from the IMPACS system, as being not worth the processing time required. Instead, a warning is printed at the end of the duty generation program whenever an unbalanced relief time occurs. This may indicate that the duty generation parameters have been wrongly specified and the scheduler may wish to take action, so that either the relief time cannot be used at all, or duties both starting and finishing spells of work at the relief time can be formed.

5.11.2 Program COMPARE

This program compares pairs of duties and deletes any duty which is included in another duty (i.e. the second duty covers all the work covered by the first) and is not cheaper than it. Under some common forms of agreement a duty can be as cheap as a similar duty containing less work: that is, if there is a minimum paid day and both duties would otherwise be paid less than the minimum, or if mealbreaks are paid through and the duties have the same spreadover. Otherwise, it may be still occasionally be possible for a longer duty to be cheaper than a shorter duty, for instance if the travel time to different relief points varies considerably.

When there is a minimum paid day, the duty generation program avoids forming two similar duties both costing the minimum if the only difference between the duties is their finishing time. Even then, however, each duty formed may still include or be included by other duties. The list below shows a set of duties in the order in which they would be formed by the duty generation program, although there would probably be other duties with second stretches on other buses formed in between these.





Duty 3 includes each of the other duties. If mealbreaks are paid, duties 1, 3 and 4 would all be paid the same and duties 1 and 4 could be deleted.

If there is a minimum paid day and duty 3 is paid the minimum, duty 2 would not have been formed, since it costs the same and can be extended without exceeding the minimum paid day. The other duties would be formed, but would cost the same as duty 3 and could be deleted.

In order to avoid having to compare every duty with every other duty, which would be too time-consuming, the COMPARE program works on batches of 50 duties at a time. When all comparisons within a batch have been made, the survivors of the first 25 duties are written to the output file. The last 25 duties become the first half of a new batch, and another 25 duties are read in. Thus, each duty is only compared with those duties formed shortly before or shortly after it. Because of the way in which the generation program operates, duties which are similar are likely to be close together in the set, and hence this is likely to find most of the duties which should be deleted.

Unless the scheduling agreement specifies a minimum paid day or paid mealbreaks, the COMPARE program is not used, because it is unlikely to eliminate many of the duties. The London Transport agreement does not have these features, and the COMPARE program in the form just described is not used. However, L.T. schedulers asked for an alternative version in which a duty is deleted whenever it is included in another duty, even if the longer duty is more expensive. This increases the risk of producing a schedule with over-cover, which then requires intervention from the scheduler to edit the duties involved, and so this version is not run automatically following the duty generation program, but the

scheduler can choose to use it if it seems appropriate, for instance if a very large number of duties has been generated.

If there is no cost incurred when work is over-covered, the standard version of the COMPARE program does not affect the minimum cost solution, and the program then performs a standard reduction of the set covering problem (e.g. see Garfinkel and Nemhauser [52]). The duties in the resulting schedule can be edited to remove any over-cover, giving a schedule equivalent to the optimum attainable from the full set of duties. However, as explained in section 6.6, the I.L.P. formulation in IMPACS includes a positive cost for over-covered pieces of work. In order to avoid over-cover, more expensive duties may be selected, so that the minimum cost solution from the reduced set of duties may be more expensive than the original optimum. This is even more likely to happen with the L.T. version of the program, since the duties kept are more expensive than those deleted. Neither version, however, affects the minimum number of duties required to cover the bus schedule and since the IMPACS formulation gives first priority to minimising the number of duties, the resulting schedule is only worsened, if at all, in the costs of the individual duties selected.

It would not be appropriate to use the COMPARE program with a set partitioning formulation, because then the duties selected would have to fit together exactly and it would no longer be true that deleting duties in this way cannot affect the minimum number of duties required to cover the schedule.

5.11.3 Program EVEN

This program rejects any duty if every piece of work which it covers is also covered by a specified number of other duties, the number being given as a parameter to the program. The idea behind this technique is that a reasonable choice of duties should be provided to cover each piece of work, by ensuring that every piece of work is covered by at least the specified number of duties (provided that the duty generation formed enough duties in the first place). However, beyond that level of coverage, extra duties are not contributing significantly to the choice available.

The duty generation program keeps a count of the number of duties covering each piece of work, partly in order to check that there are no pieces of work which have not been covered by any duty, and this information is also used by the EVEN program. The duties are considered in reverse order of a quality measure: as duties are formed, following a spell of work, the next pieces of work on other buses are considered in

ascending order of starting time. The first piece which can legally follow a mealbreak or a joinup (depending on the kind of duty being formed) gives the best duty, the next is the second best, and so on. Hence, for straight duties, the EVEN program first considers for rejection duties with long mealbreaks. Bearing in mind the possible virtues of duties which use the first available piece of work (discussed in section 5.2) all such duties are kept.

The generation of two-bus split duties is rather different in this respect, as described in section 5.8. However, as the split duties within a given range of mealbreak lengths are formed, the ordering of the next pieces of work after the mealbreak is still used to label the duties as best, second best and so on, just as for straight duties, although the labels are meaningless for the split duties. The EVEN program still treats the labelling of the split duties as a quality measure, but the effect is that the split duties are considered for rejection in a fairly random order.

In some schedules, it is essential to run the EVEN program because otherwise there would be too many duties for the ZIP program to handle; since the ZIP program holds information for all the variables in core, a maximum number of variables has to be set. At present the limit is between 5000 and 6000 variables; the exact limit depends on the relative proportions of two-bus and three-bus duties, since three-bus duties require more information to be stored.

When the number of duties is less than the maximum allowed, it is still useful to reduce the number of duties further and so reduce the time taken to run the ZIP program.

The reduction in computer time is offset by the possible suboptimality introduced by reducing the number of variables in this way; the L.P. optimum cannot be better and will probably be worse than that based on the original set of duties. The same applies to the integer optimum, although in practice, because the ZIP program does not attempt to find the integer optimum, but only a good integer solution, the solution found after reducing the number of duties will not necessarily be worse than that found using the original set.

Because of the difficulty in establishing the effect of the EVEN program on the quality of the final schedule, only the effect on the L.P. optimum has been considered. By a suitable choice of the coverage parameter, a considerable reduction in computer time can be achieved, for a slight increase in the L.P. optimum.

Table 5.1 shows the effect of the EVEN program with various values

Problem:	Full set of duties	EVEN program with coverage parameter: *100*	*75*	*50*	*25*
U147:					
Number of duties	3216 (100%)	1905 (59%)	1503 (47%)	1057 (33%)	590 (18%)
L.P. optimum	50009.9	50165.2	50334.6	50575.1	52078.8
Time (sec.)	16.3	10.9	8.0	5.7	3.3
WH69:					
Number of duties	2343 (100%)	2343 (100%)	2297 (98%)	1835 (78%)	1062 (45%)
L.P. optimum	112676.8	(Same as full	112676.8	112789.6	113487.0
Time (sec.)	19.5	set)	18.8	15.3	10.2
GM11:					
Number of duties	3385 (100%)	2593 (77%)	2135 (63%)	1640 (48%)	955 (28%)
L.P. optimum	73466.0	73466.0	73475.9	73516.4	74224.6
Time (sec.)	35.8	26.7	25.5	18.0	11.6
CA137:					
Number of duties	3226 (100%)	3226 (100%)	3137 (97%)	2440 (76%)	1350 (42%)
L.P. optimum	123285.0	(Same as full	123285.0	123351.5	123970.6
Time (sec.)	44.1	set)	45.9	36.3	20.7
AP86/b:					
Number of duties	4735 (100%)	4155 (88%)	3488 (74%)	2577 (54%)	1440 (30%)
L.P. optimum	124737.9	124798.7	124889.0	124973.1	125321.6
Time (sec.)	76.9	66.7	56.8	45.1	25.4
PM36B:					
Number of duties	7012 (100%)	6033 (86%)	4946 (71%)	3575 (51%)	1881 (27%)
L.P. optimum	(Too many to	(Too many to	169518.9	169675.4	170121.2
Time (sec.)	solve)	solve)	92.9	62.1	40.0

Table 5.1 Effect of EVEN program.

of the coverage parameter on a representative set of schedules.

For each run, the number of duties (both as an absolute number and as a percentage of the number of duties formed by the duty generation program), the L.P. optimum and the time taken to reach the optimum (on an Amdahl 5860) are given.

The reduction in computer time is very roughly proportional to the reduction in the number of duties. In fact, in one case (CA137), a slight reduction in the number of duties leads to a slight increase in the computer time: this is due to variations in the quality of the L.P. starting solution (see section 6.3). The EVEN program itself takes little time to run: for each of the examples in Table 5.1, it took only a few seconds.

The percentage reduction in the number of duties for a given coverage parameter is very variable, since it depends on the coverage pattern in the original set of duties. In small schedules with infrequent relief times, none of the pieces of work can be covered by very many duties; conversely if the schedule is large or if the relief times are frequent, there are many more possible combinations, so that each piece can be covered in many different ways and even a high value of the coverage parameter may eliminate many duties.

The effect of the EVEN program on the L.P. optimum depends on both the percentage reduction in the number of duties, and on the coverage parameter. In all six examples, a coverage parameter of 25 gives a much worse result than the others, and this might in some cases lead to a schedule with more duties than would otherwise be necessary, which would be unacceptable. Schedulers are not recommended to use a coverage parameter below 50. A coverage parameter of 50 gives acceptable results except possibly in the first problem (U147) where it reduces the number of duties by two-thirds.

The coverage parameter actually chosen in a particular case will probably depend on the original number of duties. It is of course essential to reduce the number of duties to less than the maximum that ZIP can cope with, but it is usually preferable to keep well below this limit and aim for a maximum of about 4000-4500 duties. If the original number of duties is already below this level, then it could probably be safely reduced by 25-50%, but if there are fewer than about 2000 duties originally, then probably no further reduction is necessary.

5.12 Inspecting The Set Of Duties

An inspection facility is provided in IMPACS to allow the scheduler to examine the set of duties produced. The scheduler can specify a duty

outline and the inspection program displays on the screen (in batches of ten duties at a time, if necessary) any duty which matches the outline. For instance, the scheduler can ask for all two-bus duties, or all duties starting at a particular relief time, or all duties with a specific last spell.

A common way of using this facility is to specify a complete duty which the scheduler thinks the duty generation program should have formed, to find out whether in fact it is in the set. For instance, this was useful in extending the formation of three-bus duties for London Transport: when IMPACS failed to produce the result expected, the scheduler was able to inspect the set of duties and demonstrate that a necessary three-bus duty had not been formed.

CHAPTER 6. SET COVERING FORMULATION & L.P. SOLUTION

6.1 The Set Covering Problem

The following description of the set covering problem is taken from Garfinkel and Nemhauser [52].

A cover of a set I = {1,2,....,m} is defined as follows: if P = $\{P_1,P_2,\ldots,P_n\}$, where each P_j is a subset of I for j ∈ J = {1,2,....,n}, a subset J^* of J is a cover of I if $\bigcup_{j\in J^*} P_j = I$.

If, in addition, the subsets P_j for j ∈ J^* are mutually exclusive, J^* is a partition of I.

If a cost c_j is associated with every j ∈ J, the set covering problem is to find a cover of minimum cost. The problem can be expressed as the integer linear programming problem:

$$\text{minimise} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad i=1,2,\ldots,m \qquad \text{(i)}$$

$$\text{and} \quad x_j = 0 \text{ or } 1 \qquad j=1,2,\ldots,n \qquad \text{(ii)}$$

where x_j = 1 if j is in the cover, 0 otherwise

and a_{ij} = 1 if i ∈ P_j, 0 otherwise.

As a formulation of the crew scheduling problem, the elements of the set I correspond to the pieces of work in the bus schedule, and J corresponds to the set of generated duties, so that a_{ij}=1 if the j^{th} duty covers the i^{th} piece of work and c_j is the cost associated with the j^{th} duty. The variable x_j = 1 if the j^{th} duty is in the schedule, 0 otherwise. The constraints (i) represent the fact that each piece of work must be covered by at least one duty.

The IMPACS formulation has in addition a small number of side constraints of the form:

$$\sum_{j\in J_k} x_j \quad \begin{matrix}\leq \\ \geq\end{matrix} \quad l_k \qquad \text{for } J_k \subseteq J. \qquad \text{(iii)}$$

where l_k is an upper or lower limit on the number of duties of a particular class, or on the total number of duties, if J_k = J.

The formulation also departs from the standard set covering model in introducing both slack and surplus variables on each constraint and in assigning costs to them, ~~and surplus variables,~~ as described below in section 6.5. The model therefore allows both under-cover and over-cover; this is equivalent to the formulation subsequently termed

"generalised set partitioning" by Mitra and Darby-Dowman [23], which is described in section 2.3.3.

6.2 Set Covering V. Set Partitioning

Because the constraints corresponding to pieces of work are inequalities rather than equations, integer solutions may have some constraints covered by two or more non-zero variables, which in scheduling terms would mean two or more duties assigned to the same piece of work.

In practice, the solutions produced by ZIP do not usually have over-covered pieces of work, although the likelihood of over-cover depends to some extent on the scheduler. When over-cover occurs, it most commonly results in two duties overlapping slightly; a small piece of work is assigned both to the duty covering the previous piece of work, and to the duty covering the following piece, and the scheduler must then decide which duty to delete the piece from. This kind of over-cover is more likely to occur when the scheduler restricts the duty generation process to produce duties which are individually efficient but which cannot easily be fitted together without overlapping.

Schedulers are encouraged to restrict the formation of duties so that they are as efficient as they think realistic, to keep the number of variables in the formulation to a minimum. If a set partitioning formulation were being used, the scheduler would have to bear in mind that the duties must fit together exactly. It would then be impossible to restrict the formation of duties to the same extent and the number of variables would be considerably increased.

Large amounts of over-cover, e.g. complete spells allocated to two or more duties, sometimes occur. Although this may simply mean that the solution is not good enough and should be improved, it usually reflects the conditions of the union agreement; for instance, there may be a restriction on the number of split duties so that more early and middle duties are required to cover the peaks, giving over-cover in the middle of the day. The effect of such a condition in manual schedules is that some duties have a low work content, filled up with periods of reserve or stand-by, that is, the driver reports at the depot and may be assigned to another duty if the regular driver is not available through illness, etc. This can be imitated by producing a schedule with large amounts of over-cover and editing the duties manually to give periods of reserve. Using set partitioning instead of set covering would require duties with periods

of reserve to be generated specifically: it is simpler to allow over-cover and edit the duties afterwards.

In short, although the schedule must ultimately have only one duty assigned to each piece of work, it is better to allow the formulation to include the possibility of over-cover and so to use set covering rather than set partitioning. When it occurs, over-cover can be eliminated by editing the schedule.

6.3 Equality Constraints

Although the formulation used is set covering rather than set partitioning, so that the constraints (i) are inequalities, some of the constraints are defined as equations. The corresponding pieces are the first pieces of work on early buses and the last pieces of work on late buses, 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. The identification of these pieces of work is done automatically.

The equality constraints were originally introduced to help in the construction of the starting solution (see section 6.9), and subsequently kept in the formulation because one of the branching strategies of the branch-and-bound algorithm required them. However, there are also intrinsic reasons for making these constraints equations. Although over-cover is allowed in the formulation, it can only be acceptable in the situations described in the last section, i.e. either an overlap between two spells of duty, so that the over-covered piece must be in the middle of a running board, or an over-covered spell in the middle of the day. Neither of these cases allow over-cover on the pieces of work which correspond to the equality constraints; over-cover on these pieces would necessarily indicate an inefficient solution, so that it is sensible not to allow it.

6.4 Definition Of Constraints

Each constraint in (i) corresponds to a piece of work in the bus schedule, but these are not necessarily the original pieces of work defined by the relief times. Contiguous pieces of work in the bus schedule can be combined together if the relief time separating them is not used by any duty in the generated set. If both pieces of work gave rise to a constraint in the set covering problem, then the rows of the A matrix would be identical, so that one of the constraints would be redundant.

The constraints of the set covering problem therefore correspond to

the combined pieces of work defined by the relief times which are actually used by the duties of the generated set. The combined pieces of work are renumbered and each relief time is associated with the number of the combined piece of work which follows or includes it. The list of duties, which is set up by the duty generation program in terms of the relief times starting and finishing each spell, can then be rewritten in terms of the first and last combined pieces of work in each spell, which are the same as the numbers of the first and last rows of the A matrix covered by a block of 1's in the column corresponding to the duty.

6.5 Costs Of Duty Variables

The primary objective in compiling a crew schedule is almost always to use the minimum number of duties required to cover the bus schedule. Even if the operating cost of the schedule, i.e. the total paid time of the duties, could be reduced by increasing the number of duties, the extra duties would not normally be acceptable. Given that the minimum number of duties has been achieved, minimising the operating cost of the schedule is a subsidiary objective.

In some circumstances, it may also be required that certain types of undesirable duty should be avoided, if possible. These requirements are usually unwritten rules additional to the union agreement: undesirable duties are duties which are legally valid, but for some reason are disliked by the crews or the management. The criteria defining good and bad duties are often fairly hazy. For instance, there may be a rule that "the maximum continuous driving time is 5 hours, but should preferably be less"; although the maximum is clearly stated and can be dealt with in the validation process, it is not clear what an acceptable driving time would be.

The main objective is still to minimise the number of duties, so that avoiding undesirable duties "if possible" means that they should not be used unless the total number of duties would otherwise increase. Alternatively, since there may be several different types of undesirable duty, it may be permissible to introduce an undesirable duty in order to eliminate another, even more undesirable, duty or to introduce several slightly undesirable duties in order to eliminate one very bad duty.

Some undesirable duties are also expensive, in terms of the wages paid to the crew; for instance, it is usually required that the number of split duties should be kept as low as possible, and they often attract penalty payments. Minimising the operating cost of the schedule will at the same time tend to avoid this type of undesirable duty. However, there

are many types of undesirable duty for which there is no extra payment to the crew. The requirement to avoid using undesirable duties is dealt with by attaching extra penalty costs to these duties in ZIP; the level of the penalty cost is set by the scheduler and should reflect how undesirable the duty is, compared to other kinds of undesirable duty, and perhaps how far the operating cost may be increased to improve the quality of the duties.

The objective function is designed to allow the scheduler to achieve the three aims:

a) minimise the number of duties required;

b) achieve an acceptable schedule by avoiding duties with undesirable features (if appropriate);

c) minimise the operating cost of the schedule.

The cost assigned to a duty variable in ZIP is then the sum of:

a) a large constant, specified by the scheduler;

b) any penalty costs applicable; and

c) the wages cost of the duty (in minutes paid).

ZIP produces a schedule which minimises the total cost of the duties chosen. The large constant is designed to ensure that first priority is given to minimising the number of duties in the schedule. Usually a value of 2000 is chosen; since the wages cost of a duty is usually about 450 minutes, with possibly additional allowances of up to say 300 minutes, the constant is then much greater than the wages element of the cost assigned to a variable. Moreover, since the wages cost of a duty mainly consists of the driving time, the greater part of the total wages cost of a schedule is the total driving time, which is fixed. Of the additional wages cost of each duty, a significant proportion does not vary very much according to which duty is chosen, for instance, the signing on and off time. Hence, choosing duties with low wages costs has relatively little influence on the total cost, compared with reducing the number of duties.

The constant component of the cost of a duty should also outweigh the penalty cost element, but there is some risk that by introducing an extra duty, a number of duties with high penalty costs could be eliminated and thus give a solution with lower total cost. To avoid this danger, the strategies described below in section 6.12 have been devised to achieve the aim of using penalty costs to eliminate undesirable features, as far as this can be done without increasing the number of duties required.

Although the main objective is almost always to minimise the number of duties, this assumption is not built into the ZIP program. The scheduler can increase the number of duties in the schedule above the minimum required by adding a side constraint that the sum of the variables should be at least a specified number, or, by setting the constant element of the cost of each duty to zero, it would be possible for the scheduler to minimise only the operating cost of the schedule, even if the number of duties is then higher than it would otherwise be. However, it is assumed in the branch-and-bound phase that the number of duties in the current L.P. optimum should not be exceeded, so that at that stage, no increase in the total number of duties is allowed, even if the total cost of the schedule could thereby be reduced.

6.6 Slack And Surplus Variables

For each piece of work, a slack variable and a surplus variable are added to the formulation. A non-zero slack variable means that the piece of work is not completely covered; a non-zero surplus variable means that the piece is covered by more than one duty. (For the equality constraints, surplus variables are banned).

Slack variables are generally undesirable, but in some circumstances they can fairly reasonably represent overtime pieces. For instance, when the overtime work is required over the morning or evening peak, peak-only running boards are often single pieces of work, and slack variables on these would represent acceptable overtime pieces. Overtime pieces have sometimes been dealt with in this way, but since IMPACS makes use of the fact that the L.P. optimal solution usually has no uncovered pieces so that slack variables can be banned in the branch-and-bound phase (as described in the next chapter), allowing slack variables to represent overtime pieces causes difficulties, and it is therefore preferable to form overtime pieces specifically as one-bus duties, as described in section 5.7.

The cost of a slack variable must be set at a high level, so that it is not cheaper to leave a piece of work uncovered than to bring in an extra duty. The cost is a constant, defined by the scheduler, plus the length of the piece of work in minutes. The constant should be larger than the constant component of the cost of a duty variable.

The cost of a surplus variable is calculated as the length of the piece of work, multiplied by a parameter, which is usually a small positive integer. Giving surplus variables a positive cost is usually sufficient to produce a schedule without over-cover, although in the last resort,

manual editing of the schedule may be necessary.

6.7 Side Constraints

In addition to the constraints corresponding to pieces of work, upper or lower bounds may be required on the number of duties in a specified class. Each duty when formed by the duty generation program is assigned to one of six classes corresponding to early, late, middle, split, day and overtime duties. Upper and lower limits can be specified on the number of duties in each class, on the total number of duties, on the number of uncovered pieces, or slack variables, and on the number of three-bus duties. These can be specified by the scheduler, either initially or when reoptimising an existing solution. Side constraints on the total number of duties or banning uncovered pieces are also added automatically by the program at certain points in the process, as will be described later. The total number of such constraints is always small, usually three or four at most.

6.8 Outline Of Solution Method

The problem is solved by first relaxing the constraints that the values of the variables should be integer. The resulting linear programming problem is solved and then, unless the solution happens to be already integer, an integer solution is found using a branch-and-bound algorithm.

The linear programming and branch-and-bound algorithms are contained within a package of Fortran subroutines called ZIP, which was designed to solve the set covering and set partitioning problems. It is described by Ryan in [51]. The ZIP routines consist of a set of core routines, containing the basic structure of the revised and dual simplex methods and the branch-and-bound algorithm, and a set of user routines which allow the special characteristics of the problem being solved to be built in. The user routines, for example, select entering variables for the primal and dual simplex routines, and control the branching and node-choice strategies in the branch-and-bound algorithm. The core routines in turn call a set of subroutines for handling sparse linear programming bases, which are part of the Harwell Subroutine Library; these are described by Reid [53].

The core routines were supplied by Ryan; the user routines for IMPACS were written in their original form by Ryan and the author during Ryan's Visiting Research Fellowship at the University of Leeds in 1980. The user routines have been extensively modified since then. The original

formulation has also been modified by the author, since there was at first no provision for the side constraints described in section 6.7.

The ZIP program can be run from start to finish without intervention from the scheduler, producing an integer solution suitable for conversion to a schedule. This is generally the option favoured by schedulers, at least for a first attempt. However, the program can be run in other ways: for instance, the scheduler can instruct the program to find the continuous optimum and then stop. It is also possible to pick up an existing continuous optimum, change the conditions so that it is no longer optimal (for instance, by adding a side constraint or changing the penalty costs) and reoptimise. This option is very often used, since the scheduler may want to try to improve the first schedule produced, for instance by increasing the penalty cost attached to a particular type of duty.

6.9 Starting Solution

An initial starting solution can be constructed by selecting the slack variable associated with each piece constraint. This is feasible, except possibly if there are side constraints. In scheduling terms, an all-slack solution means leaving every piece of work uncovered, which is very expensive because of the high cost of slack variables. It is easy to construct much better starting solutions, as described below.

6.9.1 Original Method

Originally a method based on the formulation described by Manington [31] was followed. This requires that some of the piece constraints should be equations rather than inequalities, the equality constraints being chosen so that no duty can be formed which covers two equality constraints simultaneously. A set of constraints with this property are the first pieces of work on early buses and the last pieces of work on late buses. Then each equality constraint is associated with a subset of duties, i.e. the duties covering the corresponding piece of work, and the subsets are mutually exclusive. A starting solution can be constructed by choosing one duty from each subset. This will ensure that each of the equality constraints is covered by exactly one duty, as required. From each subset, the duty selected is that which covers the largest number of previously uncovered pieces of work, so that when all the equality constraints are satisfied, as much as possible of the rest of the bus schedule has also been covered.

In constructing the basis, each duty variable selected replaces the slack variable on the equality constraint which it covers. The remaining

constraints may be over-covered, in which case the surplus variable enters the basis; otherwise, they are either uncovered or covered by exactly one of the selected duties, and the slack variable remains in the basis, at value zero in the latter case. This process gives a starting basis and an initial integer solution.

The method had to be modified when constraints on the number of duties of each type were introduced, since it would then sometimes result in an infeasible solution: for example, the number of duties selected is the total number of early and late buses and is smaller than the number of duties required in the schedule, so that a constraint specifying a lower limit on the total number of duties would not be satisfied. The other reason for changing the method was to improve the quality of the solutions produced.

6.9.2 Current Method

The revised procedure for constructing a starting solution uses a greedy heuristic similar to that described by Balas and Ho [54]. The pieces of work are sorted into ascending order of the number of duties covering them. They are then considered in order, and for each piece of work not yet covered, a duty covering that piece is selected according to a criterion which is dependent on the cost of the duty, the number of so far uncovered pieces which would be covered by the duty, and the over-cover which would result if the duty were selected. The process continues until all pieces of work are covered.

When there are side constraints, these have to be dealt with at the same time. The side constraints which are most commonly specified initially are upper limits on the number of duties of a particular type (usually splits) and upper or lower limits on the total number of duties. The first of these is easily dealt with, by simply checking each duty as it is added to the solution, to make sure that the upper limit on duties of that type has not already been reached. An upper limit on the total number can be dealt with similarly. A reasonable lower limit on the total number of duties is automatically satisfied, since any realistic estimate of the number of duties required in the schedule is always exceeded by the number of duties selected for the starting solution.

A lower limit on the number of duties of a particular type might conceivably be difficult to satisfy in the construction of the starting solution; if the program fails to find a feasible starting solution, the scheduler must temporarily remove the side constraints and apply them at a later stage.

If there is a realistic upper limit on the total number of duties, there will still be pieces of work uncovered in the starting solution when this limit is reached. Hence the "no uncovered pieces" constraint, which is normally used in conjunction with an upper limit on the total number of duties, is not allowed as an initial constraint.

Experiments have been carried out using several other criteria for selecting the next duty to add to the starting solution, including some of the functions suggested by Balas and Ho, and other similar functions. As reported by Balas and Ho, "none of [the evaluating functions] emerged as uniformly dominating any of the others in terms of the quality of solutions obtained." Methods for combining solutions in order to obtain an improved solution were also tried, including a method suggested by Balas and Ho, but none of the methods tried resulted in a consistent improvement, and the increase in processing time is not worthwhile.

6.9.3 Construction Of Starting Basis

In order to construct a starting basis from the heuristic solution, each duty selected is checked to see that it is not redundant, i.e. that it covers at least one piece of work which is not covered by any other duty in the solution; this is true when the duty is added to the solution, but may not remain true when later duties are added. The remaining duty variables form part of the starting basis, together with slack and surplus variables corresponding to any uncovered and over-covered pieces of work, respectively. Then for every duty variable in the basis, a surplus variable (at zero level) is added for every piece covered only by this duty, except for one arbitrarily chosen piece. This completes the basis, as far as the piece constraints are concerned, and ensures that the columns are linearly independent. If there are any side constraints, the construction of the starting solution should ensure that they are satisfied, and either the slack or the surplus variable is entered in the basis.

6.9.4 Results

The starting solutions constructed in this way usually have up to 25% more duties than the final solution, and are up to 50% more expensive. Many pieces of work are over-covered, and a considerable proportion of the extra cost is due to the cost of the over-cover rather than to the extra duties. Hence as a method of finding good heuristic solutions to the set covering problem it is not very satisfactory, but it does provide reasonably cheap starting solutions for a small amount of processing effort.

The table below shows the starting solutions found for several

London Transport schedules, with the eventual L.P. optimum in each case. For these problems, the required number of duties was known in advance, and the L.P. was constrained to use at least this number of duties.

Problem	No. of variables	No. of constraints	Starting solution *Objective*	*duties*	L.P. Optimum *Objective*	*duties*
BW25	2504	83	77319	31	62717.8	25
S72	1633	116	79090	24	50905.2	19
WH69	2481	170	127780	49	107676.2	43
NB71	1450	84	65042	21	45364.6	17
U101	3254	143	116967	39	85838.9	32
AP86	3396	213	198228	71	155643.9	62
PM36B	3575	252	203017	78	167192.7	67

6.10 Solution Of Relaxed L.P.

In linear programming problems, the number of iterations required to reach the optimal solution is normally taken to be roughly 1.5 to 3 times the number of constraints. In IMPACS, however, the number of iterations required using the standard simplex algorithm is much greater than this, probably due to degeneracy as discussed in section 6.10.3 below. A number of ways of reducing the time taken to find the continuous optimum have been investigated, some of which have been successful, and they are described below.

6.10.1 Multiple Pricing

Ryan's version of the ZIP program [51] provides the facility for multiple pricing in the primal simplex algorithm, that is on each "major" iteration, a number of profitable potential entering columns can be selected, rather than just one most profitable entering column. Each major iteration is followed by a series of minor iterations in which the entering column is selected only from this shortlist. A new major iteration is performed whenever there are no profitable columns remaining in the shortlist.

Multiple pricing is a standard technique for reducing the time taken to reach optimality. In favourable circumstances, the increased number of iterations required is more than compensated for by the reduction in the average time per iteration.

Experiments showed that choosing 10 profitable columns on each major iteration gave good results. However, the introduction of the steepest edge algorithm, described below, meant that multiple pricing was no longer sensible, and it has been superseded.

6.10.2 Steepest Edge Algorithm

The original column selection method in ZIP follows the standard simplex method and chooses the entering variable with most negative reduced cost. This is equivalent to choosing that variable which gives the greatest decrease in the objective function for a unit change in the entering variable, but ideally it would be better to take into account the accompanying changes in the other variables as well. In geometric terms, the exchange of one basic variable for another corresponds to moving from one vertex of the convex polytope defined by the linear constraints to an adjacent vertex, along an edge on which the entering variable is the only non-basic variable which is non-zero. The standard simplex method is equivalent to choosing the steepest edge in the space of the current non-basic variables, whereas in principle the best choice would be the steepest edge in the space of all the variables.

In practice, however, the extra computation involved in identifying the steepest edge may outweigh the theoretical advantage, so that although one would expect that a better choice of entering variable should lead to the optimal solution in fewer iterations, the extra time required for each iteration might cause the total time required to increase.

A practical method based on approximate calculation of the gradient of each edge has been proposed by Harris [55], and Goldfarb and Reid [56] developed a method which calculates the gradients exactly, using recurrence relations. The latter method gives good results for the problems arising in IMPACS, and the author has incorporated it into the ZIP package. The method is described in outline below.

If the A matrix is partitioned into an m × m basis matrix A_1 and a matrix of non-basic columns A_2 and we write:

$$N = \begin{pmatrix} A_1 & A_2 \\ 0 & I \end{pmatrix}$$

where I is the (n-m) × (n-m) identity matrix, then

$$N^{-1} = \begin{pmatrix} A_1^{-1} & -A_1^{-1}A_2 \\ 0 & I \end{pmatrix}$$

If η_i (i > m) is the i^{th} column of N^{-1}, the reduced cost z_i of a non-basic variable is $z_i = c^T\eta_i$.

Choosing the steepest edge in the space of all the variables is equivalent to minimising the normalised reduced cost: $c^T\eta_i / (\gamma_i)^{\frac{1}{2}}$ where $\gamma_i = \eta_i^T\eta_i$.

Explicit computation of the values γ_i for all non-basic variables at every iteration would be prohibitively expensive. The recurrence relations derived by Goldfarb and Reid allow them to be calculated accurately without an enormous increase in the amount of computation at each iteration.

If an iteration involves exchanging a non-basic column q for a basic column p, the recurrences giving the new values $\bar{\gamma}_i$ in terms of the old values γ_i are:

$$\gamma_q = 1 + \|A_1^{-1}a_q\|^2 \qquad \text{(i)}$$

$$\bar{\gamma}_q = \gamma_q / \alpha_q^2 \qquad \text{(ii)}$$

$$\bar{\gamma}_i = \gamma_i - 2\bar{\alpha}_i a_i^T A_1^{-T} A_1^{-1} a_q + \bar{\alpha}_i^2 \gamma_q \quad (i \neq q) \qquad \text{(iii)}$$

where the numbers α_i (i > m) are the components of the p^{th} row of $A_1^{-1}A_2$, and $\bar{\alpha}_i = \alpha_i / \alpha_q$.

Since (i) gives a freshly calculated value for γ_q, the accuracy of the recurrence relations can be checked by comparing this value with the existing recurred value.

The initial calculation of the weighting factors takes little time. As pointed out by Bellmore and Ratliff [57], the method described in section 6.9.3 produces a basis which by interchanging rows can be transformed into an involutory matrix (i.e. one which is its own inverse); this property is unaffected by the addition of the side constraints. Operations using the inverse of the starting basis are therefore very simple. However, if the weighting factors have to be recalculated at a later stage (for instance, if a previously-optimal basis is picked up and reoptimised with different penalty costs) the time required is much longer.

Even with the recurrence relations, the extra computation required to update the weighting factors means that each iteration takes much longer than a major iteration of the standard simplex algorithm using multiple pricing. However, for the type of problem arising in crew scheduling, the reduction in the number of iterations is such that the total time is significantly reduced.

6.10.3 Degeneracy

Degeneracy often arises in crew scheduling set covering problems, as shown by the fact that both the starting solution, constructed as described in section 6.9, and the final integer solution have far fewer duties than there are pieces of work to be covered, so that most of the variables in the starting basis and in the solution basis must be zero.

A perturbation method for dealing with degeneracy, described by Benichou, Gauthier, Hentges and Ribiere [58] was introduced by the author into the primal simplex method in ZIP, partly because Benichou et al. suggested that degeneracy causes considerable increases in solution time, and partly to avoid the possibility of cycling problems. Only one case has occurred in which cycling was suspected: no improvement in the starting objective was found for many thousands of iterations when multiple pricing was used; without multiple pricing, a solution was found without undue difficulty, although there was no improvement for the first few hundred iterations. Investigation showed that initially no individual non-basic variable could cause a reduction in the objective. It appeared that at first there was fairly random replacement of one zero basic variable by another, which eventually allowed a real change to occur.

In the perturbation method proposed by Benichou et al., whenever degeneracy is detected, every zero basic variable corresponding to an inequality constraint is perturbed by adding a constant ε to it. The current basis then becomes a feasible solution to a perturbation of the original problem, in which the right hand side has added to it ε times the sum of the column vectors corresponding to the perturbed basic variables.

The perturbed problem is then solved, and perturbed again if degeneracy recurs. At the end, the optimal solution for the perturbed problem is dual feasible for the original problem, but not necessarily optimal; in practice, the solution is optimal for the original problem more often than not, and otherwise only a few dual iterations are required to restore optimality.

In spite of the claims for this method, when implemented in ZIP it was found that it did not reduce solution time, and could easily increase solution time unless the perturbation was applied only very sparingly, for instance, when no change in the objective had occurred for a hundred iterations. Eventually, it was dropped from ZIP, and no other case of suspected cycling has been found.

6.11 Adjusting L.P. Optimum Solution

Sometimes the solution to the relaxed L.P. is naturally integer, and can be converted immediately into a practicable schedule. Otherwise, an integer solution must be found by branch-and-bound, as described in the next chapter.

Before entering the branch-and-bound phase, the program adjusts the L.P. optimum if necessary to ensure that the sum of the duty variables is integral. An integer solution must of course have an integral number of duties. Because the objective function is so heavily weighted towards minimising the total number of duties, it has also been found that (at least in the basic case in which the cost of a duty variable is a large constant plus the wages cost of the duty) the number of duties in an integer solution is very rarely less than the sum of the duty variables in the L.P. optimum, and then only marginally so. For instance, if the sum of the duty variables in the L.P. optimum were 66.01 then it might be possible to find an integer solution with only 66 duties, but if the sum were 66.5, any integer solution would have at least 67 duties; it would then be worthwhile to add a constraint that the sum of the duty variables should be at least 67, and find a new L.P. optimum using the dual simplex algorithm. This reduces the gap between the L.P. and integer optimal objectives and so expedites the branch-and-bound search.

When the sum of the duty variables in the L.P. optimum is fractional, the program automatically adds a constraint rounding the sum of the duty variables up to the next integer and reoptimises before entering the branch-and-bound phase, and no intervention is required from the user.

(At present, the program does not check that the sum of the duty variables cannot be rounded down, even if it is only slightly greater than the next lower integer (e.g. 66.01 in the example above). In such a case it would be up to the scheduler to check, by picking up the L.P. optimum solution and adding the appropriate side constraints (e.g. that the sum of the duty variables should be less than or equal to 66 and that no work should be left uncovered).)

6.12 Treatment Of Penalty Costs

In the case where penalty costs are used in order to discourage several different kinds of undesirable duty, and the penalty costs are large, it may no longer be true that the cheapest solution has the minimum number of duties. However, the intention in using penalty costs is to allow undesirable duties to be used if the schedule would

otherwise require more duties.

There are two alternative strategies built into IMPACS for obtaining a schedule with the minimum number of duties when penalty costs are employed, depending on whether or not the scheduler can specify in advance the number of duties required in the schedule. As described in chapter 9 and the Appendix, the scheduler specifies in a parameter file which of the various possible ways of running the ZIP program is to be used. When the scheduler chooses to use penalty costs and to run the ZIP program without intervention from start to finish, the parameter file must also contain another value which should be either the number of duties required in the schedule, or 0, indicating that there is no target number of duties.

When the scheduler does not know the required number of duties in advance, the L.P. is first solved without any penalty costs, in order to establish the minimum number of duties attainable in an integer solution (i.e. the sum of the duty variables in the L.P. optimum, rounded up to the next integer if fractional). The program then adds a new side constraint, that this minimum number of duties should not be exceeded. Since it is always possible to cover a schedule in a specified number of duties, by not covering all the work, this constraint must be combined with a constraint that no work should be left uncovered. The costs of the duty variables are recalculated to include the penalty costs. The previous L.P. optimum is used as a feasible starting solution and a new optimum is found, from which an integer solution and hence a schedule with the correct number of duties can be derived.

On the other hand, when the required number of duties in the schedule can be specified in advance by the scheduler, the penalty costs are included in the cost of each duty from the start. In practice, the addition of penalty costs will not usually cause the target number of duties to be exceeded in the minimum cost solution, and the relaxed L.P. solution will have the correct sum of variables: in that case, by making use of the fact that the minimum number of duties is known in advance, the reoptimisation required in the alternative strategy is avoided. In cases where the sum of the duty variables in the L.P. optimum does exceed the target number, the dual simplex algorithm is used to reduce the total to the target value, if possible; new constraints, that the total number of duties should be less than or equal to the target and that no work should be left uncovered, are added. If it is only the penalty costs which have caused the target number of duties to be exceeded, a new optimum will be found, but if the dual simplex algorithm fails to find a

feasible solution, the target number of duties is unattainable from the current set of variables, irrespective of the penalty costs; the action to be taken in that case is discussed in the next section.

6.13 Failure To Find A Feasible Solution.

There is always a feasible solution to the relaxed L.P. problem, unless the scheduler has specified a target number of duties. In that case, if it is impossible to select a schedule from the current set of variables with no more than this number of duties, without leaving work uncovered, the program prints the message "THE TARGET NUMBER OF DUTIES CANNOT BE MET" and stops. The scheduler must than take appropriate action and try again.

The option of specifying a target number of duties was developed for London Transport, and so far has been used only by L.T. schedulers, so that in fact the possibility that there is no feasible solution to the relaxed L.P. does not arise for other users of IMPACS.

Because the ZIP program will fail to find a solution if the target number of duties is unattainable, a target should only be specified when it would be unacceptable to use more duties. However, the target may represent an intention rather than an expectation: London Transport schedulers usually specify a target number of duties, but if the program fails to find a feasible solution, this may be taken as an indication that the target is over-optimistic for the current bus schedule. The scheduler may then remove some work from the bus schedule before generating a new set of duties and trying again with the same target number of duties.

On the other hand, if the scheduler feels that it should be possible to cover the bus schedule as it stands within the target number of duties, the set of variables must be enlarged, to give the ZIP program a wider selection, before trying again.

CHAPTER 7. BRANCH-AND-BOUND

7.1 Introduction

Finding an integer solution when the solution to the relaxed L.P. problem is non-integer can be time-consuming and difficult. Considerable efforts have been devoted to tailoring the branch-and-bound algorithm to the characteristics of the crew scheduling problem. Since the simplifications already made in the earlier stages of the process mean that an optimal integer solution to the problem as formulated would not necessarily be an optimal schedule, the emphasis is on finding a good integer solution as quickly as possible, rather than on finding the integer optimum and proving optimality.

7.2 Additional Constraints

As explained in section 6.11, by the time that the branch-and-bound phase is entered, the L.P. optimum has been adjusted if necessary so that the sum of the duty variables is an integer, N. At this point, side constraints are added to ensure that the total number of duties cannot exceed N and that no work can be left uncovered (provided that these constraints have not already been added).

Without these constraints, nodes may appear in the branch-and-bound tree at which the sum of the duty variables is N+f, where f is a small fraction ($0<f<1$). Such a node is not worth pursuing, although it may temporarily appear to be so, because by the same argument as in section 6.11, any integer solution found further down the tree will have N+1 duties (unless f is very small, and the nodes further down the tree revert to N as the sum of the duty variables, which is very unlikely). If the sum of the duty variables in the L.P. optimum is N, experience shows that it is more than likely that an integer solution with N duties can be found, and a solution with N+1 duties would then be unacceptable.

With the additional side constraints, every node in the branch-and-bound tree has the sum of duty variables equal to N, with no work uncovered. If it turns out that no integer solution can be found with N duties, it may then be necessary to look for solutions with N+1 duties, but in that case a new branch-and-bound tree should be formed in which the sum of the duty variables at every node is N+1. This is discussed below in section 7.14.

7.3 Reduction In Problem Size

As explained in the last section, any integer solution found must have the same number of duties as the current L.P. optimum. A heuristic

reduction technique has been developed, based on the expectation that there are at least some good integer solutions which are similar to the L.P. optimum in other ways, in particular that the way in which the fractional duties in the continuous solution cover the bus schedule gives an indication of how the bus schedule might be covered in an integer solution.

The duties corresponding to the non-zero variables in the L.P. optimum use only a subset of the relief times in the bus schedule, and the duties in an integer solution will use a still smaller number of relief times. The reduction technique assumes that an acceptable integer solution can be found by restricting the choice of relief times to the subset used by the duties in the L.P. optimum, so that any duty which uses a relief time not in this subset can be excluded from the search for an integer solution.

Although there is no theoretical justification for this assumption, it seems to be reasonable in practice, except in a very few instances. The most serious danger is that the only integer solutions with the required number of duties use one or more relief times outside the initial subset, in which case the use of the reduction technique would cause a failure to find an integer solution, whereas without it one might be found. This has happened on occasion, but usually there are many different possible integer solutions, and it would then be very unlikely that the reduction technique should exclude them all.

A more likely danger is that a better integer solution could be found amongst the complete set of duties than amongst the reduced set. However, as described in section 7.5, the search terminates as soon as a good integer solution is found, so that if a worse solution were found using the reduced set than with the full set, it would not necessarily indicate that the unknown integer optimum had been excluded. Indeed, it can happen that the solution found using the reduced set is better than that using the full set.

The result of banning any duty which uses a relief time outside the initial subset is that the number of variables is much smaller than in the original problem. Usually more than half of the duties are banned, which means that the time taken to solve each node is very much reduced. A reduction in the total solution time cannot be guaranteed, since the branch-and-bound tree will develop differently and the number of nodes to be evaluated could possibly increase, but in practice there is usually a considerable reduction.

In the case where the integer solution from the reduced set of

duties is worse than that found using the full set, the two solutions must have the same number of duties, because of the additional constraints placed on the problem. The solutions differ only in the cost and quality of the duties selected. It is possible that the heuristic improvement techniques described in chapter 8 may be able to eliminate some of these differences.

7.3.1 Results

The effect of the reduction technique on solution time and quality of schedule for a sample set of London Transport problems is shown in Table 7.1. The "cost of the schedule" shown there is the total cost of the duties after the improvement techniques described in chapter 8 have been applied. In one case, problem TL47, both the L.P. optimum and the integer solution contained over-cover. The duties were edited to produce a schedule with no over-cover, which explains why the cost of the TL47 schedule is less than the L.P. optimum.

The sample problems are arranged in ascending order of difficulty in finding an integer solution, as measured by the number of developed nodes in the tree at termination. In the simplest cases, an acceptable integer solution is found very quickly, whether or not the set of eligible variables is reduced, and the solutions found are identical. For the middle range of problems, reducing the number of variables leads to considerable savings in computer time; not only is each node solved more quickly, but fewer nodes need to be solved before a good integer solution is found.

The last two problems are difficult to solve: the only integer solutions found are not sufficiently good to halt the tree search, which only terminates when 50 nodes have been developed. In one of these problems, using the full set of variables gives a slightly better solution, though at the expense of an increase in computer time; on the other hand, in the final problem, no integer solution can be found in 50 nodes with the full set of duties.

In only one case does the full set of duties give a significantly better solution, whereas in three cases, the reduced set of duties gives a cheaper solution.

These sample problems illustrate the general conclusions based on experience with this reduction method:

a) the computer time required to find a good integer solution is very much reduced, both because each node can be solved more quickly and because often a solution is found in fewer nodes;

Problem	CA37	S72	AP86/b	GM11	TL47	PM36B	AP86/a	WH69	HL207
No. of duties in schedule	38	19	50	27	39	67	62	43	32
No. of variables	2467	1887	3489	2593	1633	3575	5039	2195	2993
No. of piece-constraints	186	97	226	159	165	251	235	169	159
L.P. optimum	102561.1	51581.0	124889.5	73591.3	100186.7	169675.4	159928.2	112726.7	85877.3
Reduced set of duties:									
% excluded	67	73	73	69	49	54	59	53	72
No. of nodes in tree	2	3	8	9	12	13	23	50	50
Time (sec.)	2.3	0.6	8.2	22.2	13.7	48.9	29.8	65.1	70.2
Best integer found	102605	51690	124961	74305	100965	170122	160183	114527	86552
Cost of schedule	102605	51690	124961	74305	99022	169977	160142	114527	86448
Full set of duties:									
No. of nodes in tree	2	3	20	41	26	32	34	50	50
Time (sec.)	4.7	3.1	106.4	153.5	24.6	208.0	198.7	105.3	143.0
Best integer found	102605	51690	125012	74305	100484	169976	160402	114437	None
Cost of schedule	102605	51690	125012	74305	99046	169976	160285	114437	None

Table 7.1. Effect of reduction in problem size.

b) the quality of the solutions found is not consistently better or worse;

c) the risk of not finding an acceptable integer solution is probably less than if the full set of duties is used.

7.4 Branching Strategies.

The basic framework of the branch-and-bound algorithm is determined by the ZIP routines. As each node is solved, then unless the node is fathomed, the tree branches to two new subproblems. The next node to be solved, whether or not the node just solved is fathomed, is selected from the set of all active nodes.

Within this framework, there is provision for special-purpose routines to override the ZIP branching strategy and choice of next node to solve.

If this facility is not required, the default choices made by ZIP can be accepted instead. The defaults are that at a node with a fractional solution, that variable closest to an integer value (0 or 1) is selected as the branching variable. The default choice of node to be solved is the one whose parent node has least objective value; the one branch from the parent node is chosen if it has not yet been solved, otherwise the zero branch. Other strategies which have been tried will be described in sections 7.8 to 7.11.

7.5 Termination

In the crew scheduling application, it is sufficient to find a good solution, not necessarily the optimal solution. As soon as an integer solution is found, any active node whose value is within a specified percentage of the integer objective is fathomed. The percentage is an input parameter to the program and is usually 0.5%. In most cases, the first integer solution found fathoms all of the active nodes of the tree and hence terminates the branch-and-bound process; often the L.P. optimum is itself within 0.5% of the first integer solution.

Even if the search does not terminate immediately, any subsequent improvement must be cheaper than the incumbent solution by at least 0.5%, and it is unusual for the program to find such an improvement.

Since the total number of duties is constrained to be the same as in the L.P. optimum at all nodes of the branch-and-bound tree, the difference in cost between a good integer solution and the unknown integer optimum represents the use of more expensive duties and possibly less desirable duties.

7.6 Integer Allocation

In the original version of the program, a heuristic method was used to construct an integer feasible solution at any fractional node of the branch-and-bound tree. A "greedy" algorithm was used in which the basic variables were considered in order of integer infeasibility (i.e. the difference between the value of the variable and the nearest integer) and forced to take integer values of 0 or 1, until either a cover was achieved or the previous best integer bound was exceeded. The integer solutions found in this way had two functions: first, they provided good integer bounds, so that some nodes of the tree could be shown to be unprofitable and remain unsolved, and secondly, an integer solution found in this way was often good enough to fathom all the remaining active nodes and hence terminate the tree search.

However, the L.P. solution is now always rounded up to the next integer number of duties by adding a constraint on the total number of duties, with the specific intention of reducing the gap between the continuous and integer optima. This removes much of the room for manoeuvre required by the heuristic, and it is very unlikely to find a cover without using more duties, or leaving work uncovered, both of which would be unacceptable and are in any case banned by additional constraints in the branch-and-bound phase. The use of the heuristic has therefore been abandoned. Its function in fathoming unprofitable nodes is now in a sense replaced by the upper limit on the sum of the duty variables: some nodes are infeasible because it is not possible to find solutions within the specified number of duties.

7.7 Limits On Tree Size

Since the initial basis must be stored for any developed node which has an unsolved node branching from it, as must some additional information for every developed node in the tree, a considerable amount of storage space is required, and a maximum size for the tree must be specified. The maximum number of developed nodes has been set at 50, and the maximum number of bases which can be stored at 40; hence if no integer solution can be found within these limits, no schedule can be produced. It sometimes happens that an integer solution is found with a value which is too high to fathom all of the active nodes, and the program may then continue to build the tree until the limits are reached: the best integer solution found must then be accepted. In the great majority of cases, however, an integer solution can be found well within the limits allowed and is sufficiently good to halt the search immediately.

Storage limits are fairly arbitrary, but another consideration is the time required to build a tree of the maximum size allowed. If an acceptable integer solution has already been found but is not sufficiently good to terminate the tree search, there is little point in spending time building a larger tree; on the other hand, if no integer solution has been found in 50 nodes, then as pointed out in section 7.14 below, this is usually a sign that the program is in difficulties and again allowing more time to build a larger tree is unlikely to be productive. From this point of view, a maximum tree size of 50 nodes seems to be a reasonable choice.

The storage limit terminates the search when a node which has just been solved would be the 51st branching node or space is required to store the 41st basis. If storage space were the main consideration, it would be possible to evaluate the existing unsolved nodes, without forming any more branches, in the hope that an integer solution could be found. However, this would take a long time, and would not be very likely to produce an integer solution.

7.8 Constraint Branching

At a very early stage, it was found that the variable branching strategy built into ZIP as the default was not satisfactory for the crew scheduling problem. The zero branch, where an existing fractional basic variable is forced to take the value 0, is usually ineffective, since out of the whole set of variables there will be several which are very similar to the variable being banned, and so it is often easy to find a new solution without the banned variable. The one branch, on the other hand, may require a considerable change in the solution, taking many iterations and resulting in a large increase in the objective value. Since the ZIP default is to choose the node with smallest objective value to solve next, the tree tends to solve the nodes formed along a series of zero branches, with little progress towards integrality.

In order to produce a more evenly-balanced tree, in which both branches would tend to have a similar effect on the objective, Ryan developed a strategy, which he called constraint branching, for choosing the branch to be formed at a node: this is described in the ZIP report [51] and in [26]. In effect, each branch involves a set of variables rather than the single variable of the default.

Ryan developed the idea of constraint branching for set partitioning models, and the argument requires at least some of the constraints to be equalities. In theory, constraint branching could lead to suboptimality if applied to inequality constraints, but since the IMPACS formulation has

equality constraints corresponding to the first pieces of work on early buses and the last pieces on late buses, the strategy can still be applied without loss of optimality.

In a fractional solution, there must be a pair of piece constraints such that the sum of those variables which cover both constraints is strictly fractional. The remaining fraction of cover is provided by variables which cover one constraint but not both.

This can be shown by considering a variable which is strictly fractional and covers an equality constraint r_1. Then there must be at least one other non-zero variable which also covers r_1. Let r_2 be a constraint which is covered by one of these duties but not the other; suppose that d covers r_1 and r_2, and d' covers r_1 but not r_2. (r_2 need not be an equality constraint.)

If $J(r_1)$ is the set of variables covering r_1, $J(r_1,r_2)$ is the set of variables covering r_1 and r_2, and $J(r_1,\bar{r}_2)$ is the set of variables covering r_1 but not r_2, then since r_1 is an equality constraint:

$$\sum_{j \in J(r_1)} x_j = 1$$

and

$$\sum_{j \in J(r_1,r_2)} x_j + \sum_{j \in J(r_1,\bar{r}_2)} x_j = 1$$

Since $d \in J(r_1,r_2)$ and x_d is strictly fractional:

$$\sum_{j \in J(r_1,r_2)} x_j > 0$$

Similarly, since $d' \in J(r_1,\bar{r}_2)$ and $x_{d'}$ is strictly fractional:

$$\sum_{j \in J(r_1,\bar{r}_2)} x_j > 0$$

so that

$$0 < \sum_{j \in J(r_1,r_2)} x_j < 1$$

and hence r_1 and r_2 form a pair of piece constraints such that the sum of the variables covering both constraints is strictly fractional.

The equivalent of the zero branch and the one branch are respectively:

$$\sum_{j \in J(r_1,r_2)} x_j \leq 0$$

and

$$\sum_{j \in J(r_1,r_2)} x_j \geq 1$$

Along the zero branch, all duties covering both constraints at once are banned. Since r_1 is an equality, the one branch implies that any variable which covers r_1 but not r_2 must be excluded.

Hence the two branches can easily be imposed by banning a set of variables in each case: those covering both r_1 and r_2 on the zero branch, and those covering r_1 but not r_2 on the one branch.

In IMPACS it very occasionally happens that a node of the branch-and-bound tree has a fractional solution but no constraint branch can be formed, because none of the equality constraints can form part of a pair of constraints with fractional cover. In this case, the program reverts to the ZIP default variable branch, but in practice an integer solution is always found very quickly.

The required equality constraints are the first pieces of work on early buses and the last pieces on late buses, as described in section 6.3. It is largely fortuitous that these particular constraints are used to form the constraint branches; [26] describes constraint branching for set partitioning problems, in which any piece constraint can be used to form a constraint branch, and it is believed that Ryan did not intend that the constraints should be considered in any particular order when forming a branch. However, the first and last pieces of work on a bus are good choices to form constraint branches. Pieces of work in the middle of the day can be associated with many other pieces of work when forming duties; unlike the very early and very late pieces of work, they can both precede and follow other work in a duty, and if split duties are allowed the choice is even wider: a piece of work over the evening peak, for instance, could be associated with a morning peak piece of work or with a late evening piece, or many others in between. The large number of possibilities for the constraint r_2 would tend to make the zero branch relatively weak, and produce a similarly unbalanced tree to the variable branching strategy.

Forming constraint branches on the very early and very late pieces of work is, however, a successful branching strategy. Fixing the characteristics of the duty covering one of the equality constraints has a considerable influence over the rest of the schedule, and hence as well as giving an even branch in terms of the change in the objective function, it also gives an impetus towards integrality.

7.9 Node Choice

As well as overriding the default branches from each node, Ryan suggested changing the choice of next node to solve. Since the aim is to

find a good integer solution as quickly as possible, the sum of the integer infeasibilities at the parent node is taken into account as well as the objective, where the integer infeasibilities are the differences between each fractional variable and the nearest integer. The reason for this is that the nodes at which the sum of the integer infeasibilities is small seem most likely to lead to integer solutions.

In the default node choice, the next node is chosen simply on the basis of the objective value at the parent node. However, when devising a measure which combines the sum of the integer infeasibilities with the node objective, it is the increase in the objective compared with the L.P. optimum which is important. In fact, the changes in node objective throughout the tree are of relatively small magnitude compared to the L.P. optimum, particularly since the side constraints were introduced which ensure that the sum of the duty variables is the same at every node.

For each active node in the tree, the value of $R = fV$ is calculated for the parent node, where f is the sum of integer infeasibilities. V is the node objective less 99.5% of the L.P. optimal objective Z: since at the root of the tree, the node objective is the same as Z, V is calculated in this way to ensure that it is never zero. The parent node for which R is minimum is chosen, and as in the default node choice, the one branch from this node is chosen if it has not already been solved, and otherwise the zero branch.

Since it often happens that one or other of the dependent nodes has a smaller sum of infeasibilities than the parent node, with only a small increase in the objective value, this node choice strategy tends to lead to a "depth first" search, with little backtracking.

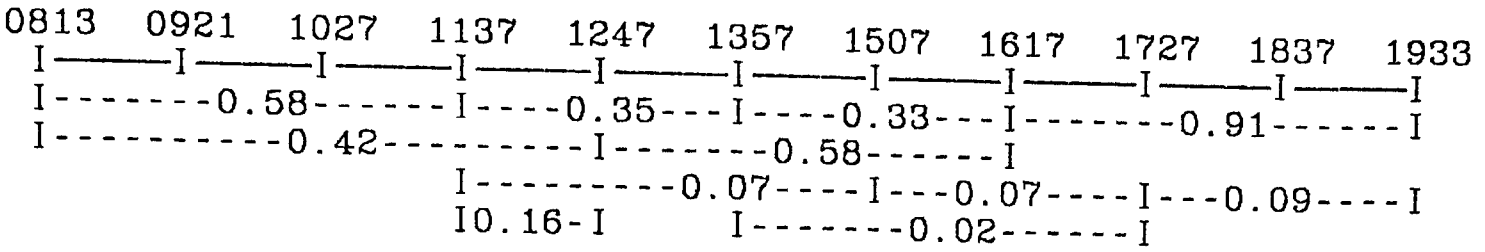
7.10 Relief Time Branching

The constraint branching strategy was used for much of the work described in chapter 9 and was reasonably successful, certainly when compared with the standard variable branching strategy which it replaced. However, the time taken to find an integer solution was still in many cases very long, and in a small proportion of cases no integer solution could be found within the limits set.

For instance, the particular schedule which caused the author to reconsider the branch-and-bound strategy was a small London Transport schedule requiring 17 duties. It was known that a 17-duty schedule could be found, because the purpose of the exercise was to improve the existing schedule. Further, the scheduler checked that the 17 duties comprising this schedule were in fact included in the set of duty variables. However,

no integer solution could be found within 50 nodes using constraint branching.

The diagram below shows the coverage of a running board in the continuous L.P. solution for this schedule. The dashed lines represent spells of duty; the numbers associated with them are the values of the corresponding variables.



It should be noted that every piece of work on the running board is covered exactly once, i.e. there is no over-cover. Every duty shown as covering part of this bus also has at least one other spell of duty on another bus, so that this complicated coverage pattern extends over the entire bus schedule.

Because there is no over-cover on this bus, the sum of the variables corresponding to duties which finish a spell of duty at a particular relief time is equal to the sum of the variables starting a spell of duty then: in the example, the sums are 0.58 for the relief times 1137 and 1247, 0.35 for 1357 and so on. If we define the changeover at any relief time as the sum of the variables which finish (or start) a spell of duty at that time, then, ignoring the possibility of over-cover, the changeovers have values between 0 and 1, and a necessary condition for the solution to the original problem to be integral is that they should have integral values. (In practice, this is probably also a sufficient condition for integrality, although counterexamples can be constructed.)

Considering the fractional solutions in this way leads to the idea of forming branches on relief times, by forcing fractional changeovers to take values of 1 or 0.

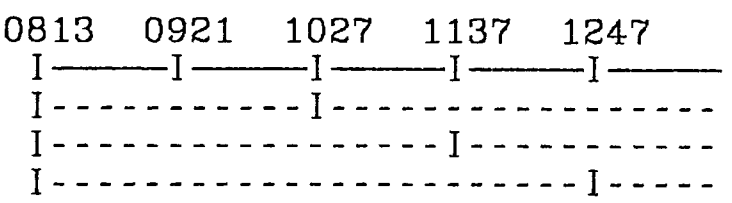
As in constraint branching, this can be done by banning a set of variables on each branch. For instance, the relief time at 1137 in the example shown has a fractional changeover with value 0.58. In an integer solution, the changeover at this time must be either 0 or 1, or in other words, either the same duty covers the work before and after this relief time, or one duty finishes a spell of duty then and another duty takes over. Hence, on one branch, all duties which start or finish a spell of duty at 1137 are banned; on the other, all duties which work on this bus from 1027 to 1247 are banned.

Because of the reduction technique described in section 7.3, any relief time which is not used by the non-zero basic variables in the L.P. optimum cannot subsequently reappear, even if there are duties in the original set of variables which use that relief time. In the example, any duty which uses the relief time at 1027 will have been banned, so that the first spell on this bus can only finish at 1137 or 1247.

7.10.1 Choice Of Relief Times

The relief times chosen to form branches are those with fractional changeovers towards the beginning and end of each bus. The buses are first sorted in ascending order of starting time and descending order of finishing time. When forming a branch, the buses are considered in sorted order: alternately the next earliest starting time, then the next latest finishing time, and so on.

When considering an early starting bus, a potential branch can be formed on the first changeover, if this is fractional. The second changeover is also considered, and the branch is not formed on this bus unless the sum of the two changeovers is sufficiently large. If a significant fraction of the cover at the beginning of the bus is provided by duties changing over at a third relief time, as shown in the diagram, the branch should not be formed.



Branching on one of only two fractional changeovers means that forcing the first changeover to 0 is likely to have the effect of forcing the second to 1, and vice versa, whereas if three fractional changeovers are involved this would not be the case. The rule is applied in reverse on the late buses.

The buses are considered in the order described because typically a duty which starts by taking an early bus out of the garage has its second spell of duty, following the mealbreak, taking over a later starting early bus from its first driver. The pattern of the late duties is similar in reverse. Hence the earliest starting duties restrict the way in which later starting duties can be formed, and should be formed first. For the same reason, this is usually the way in which manual schedulers and heuristic crew scheduling systems work.

So far, the argument has ignored the possibility of over-cover. Because over-cover can exist, it is possible in an integer solution to have

a duty starting or finishing at a relief time, as well as a duty covering the work both before and after the relief time. One branch bans the first of these duties, and the other the second, and so to that extent, the relief time branching strategy may exclude an optimal solution, although the effect of the suboptimality is not likely to be serious. If the continuous solution at a node has over-cover on either of the pieces preceding or following a relief time, that relief time is not used to form a branch.

When the first or last piece of work on a bus is an equality constraint, this form of branching is in some respects similar to constraint branching: suppose that some duties covering the first piece of work, p_i, on a bus also cover the following piece, p_{i+1}, whereas other duties do not, but instead have a spell of duty on another bus. Then a constraint branch on pieces p_i, p_{i+1} would be equivalent to a branch on the intervening relief time.

7.11 Node Choice With Relief Time Branching

The fractional changeover at a relief time can be thought of as the probability that this relief time will be used as a changeover time in an integer solution; in the example, it seems much more likely that 1617 will be used as a changeover time than 1727 (fractional changeovers 0.91 and 0.09 respectively). It would seem more productive to force the changeover at 1727 to 0 than to 1. Hence, of the two nodes branching from any developed node, the first to be evaluated is always that which forces the fractional changeover at the branching relief time to whichever is the nearer value of 0 and 1.

The difference between the fractional changeover at the parent node and the value which it is forced to take along a branch can also be thought of as an indication of the change in the objective which is likely to result when a node is evaluated. For instance, if a fractional changeover has a value close to 1, one would expect that forcing the changeover to 1 would result in a small change in the objective, whereas forcing the same changeover to 0 would be likely to result in a large increase in the objective. Since the second branching node is the one which forces the greater change in the fractional changeover at the branching relief times, it is likely to cause the greater increase in objective value and also take longer to solve, particularly if the fractional changeover is close to 0 or 1. However, if the node choice strategy is based on the objective value at the parent node when choosing the next node to evaluate, the second branch from a parent node looks as attractive as the first; it would be better, if possible, to judge candidate nodes according to the estimated objective value which would result if

they were solved.

The objective value is estimated by calculating the average increase in objective value for a unit change in the fractional changeover, based on the experience in the tree so far. Then the estimated objective value of a candidate node is the objective value of the parent node plus the expected increase due to the change in fractional changeover. The estimation is necessarily crude, but it is sufficiently discriminating to distinguish between nodes which are likely to lead to a small change in objective and can be solved quickly and those likely to lead to a very large change, which will take a long time to solve.

As with constraint branching, the choice of next node is also based on the sum of integer infeasibilities, f, at the parent node. For each active node in the tree, the value of $R = f^2 V$ is calculated, where V is calculated as described in section 7.9, except that the estimated objective value is used, rather than the actual objective value at the parent node. The node for which the value of R is least is chosen. This choice strategy gives more weight to the sum of integer infeasibilities than that used in constraint branching.

7.12 Tree Development

Unlike the constraint branching strategy, relief time branching is not intended to produce an evenly-balanced tree. In fact, the method exploits the fact that the tree is unbalanced, in that at most nodes one branch is likely to be much more attractive than the other, in order to avoid evaluating both branching nodes.

The node choice strategy tends to bias the tree development towards a "depth first" search, since the sum of integer infeasibilities at the nodes along a sequence of branches decreases fairly steadily. Even if there is a slight increase in the sum of integer infeasibilities, the program will usually favour evaluating a node branching from the last node developed rather than backtracking, provided that there has been only a moderate increase in the objective compared with the parent node, since the estimated increase in objective for this node will often be much less than the estimated increase for the active nodes further back up the tree.

This contrasts with the unbalanced tree produced by variable branching. In that case, the one branch tends to give a large increase in the objective and the zero branch a small increase, so that the tree tends to develop along a series of zero branches, but there is no particular tendency towards integrality along the zero branch.

Figures 7.1, 7.2 and 7.3 show the branch-and-bound trees formed for the same problem by variable branching, constraint branching and relief time branching, respectively, which demonstrate the typical tree development for each strategy. The nodes are numbered in the order in which they are developed. To the right of each node is shown the objective value and (in brackets) the sum of integer infeasibilities. For each node, the branch to the right is the first to be developed (the one branch in variable or constraint branching).

The variable branching tree continues to grow as shown in Figure 7.1 until the tree contains 50 developed nodes, at which point the search terminates without finding an integer solution. The zero branch makes no significant change to the objective value, or to the sum of integer infeasibilities, whereas the one branch often causes an increase in both, so that the tree develops along a series of zero branches as shown. The tree would only develop from the node at the end of a one branch if there were a decrease in the sum of integer infeasibilities with only a modest increase in the objective values, but this does not happen before the search is terminated. The time taken to build the tree thus far is 136 sec., most of which is required to evaluate the expensive one branches; the zero branches are evaluated in only one or two iterations.

The constraint branching tree develops in a more evenly-balanced way, as intended. At most nodes, either both or neither of the branching nodes are solved, and the resulting increases in objective value are of similar magnitude along each branch. The first integer solution found is 170760, at node 31. The L.P. optimum is within 0.5% of this value, so all remaining active nodes are immediately fathomed and the search stops; the total computer time is 92 seconds.

The relief time branching strategy finds an integer solution with value 170368 after developing 13 nodes and takes 29 sec. Since the original L.P. optimum is within 0.5% of this integer value, all the remaining active nodes are fathomed and the search terminates immediately. As can be seen, the second branching node is only evaluated at node 12, where the first branch results in a considerable increase in the sum of integer infeasibilities.

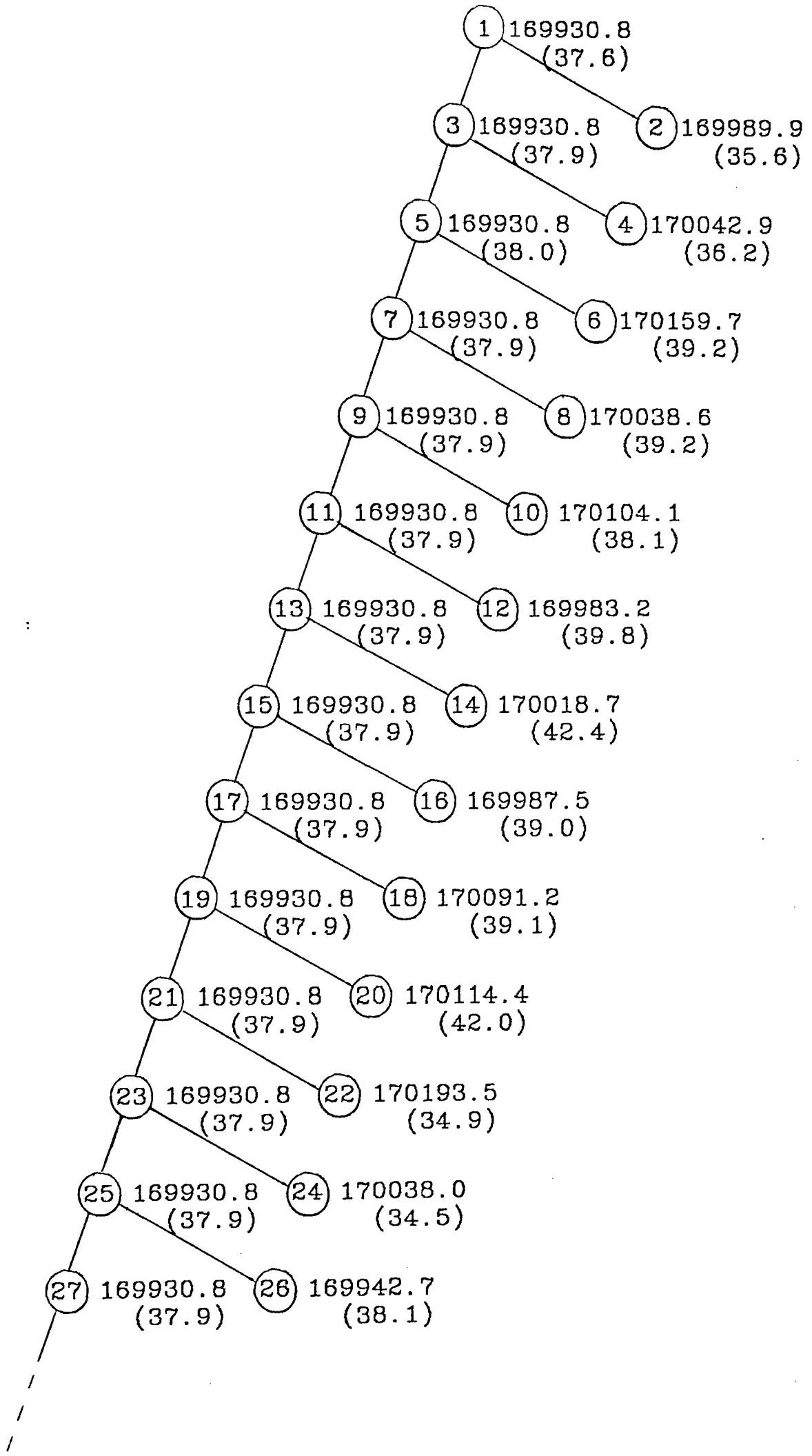


Figure 7.1. Example of variable branching

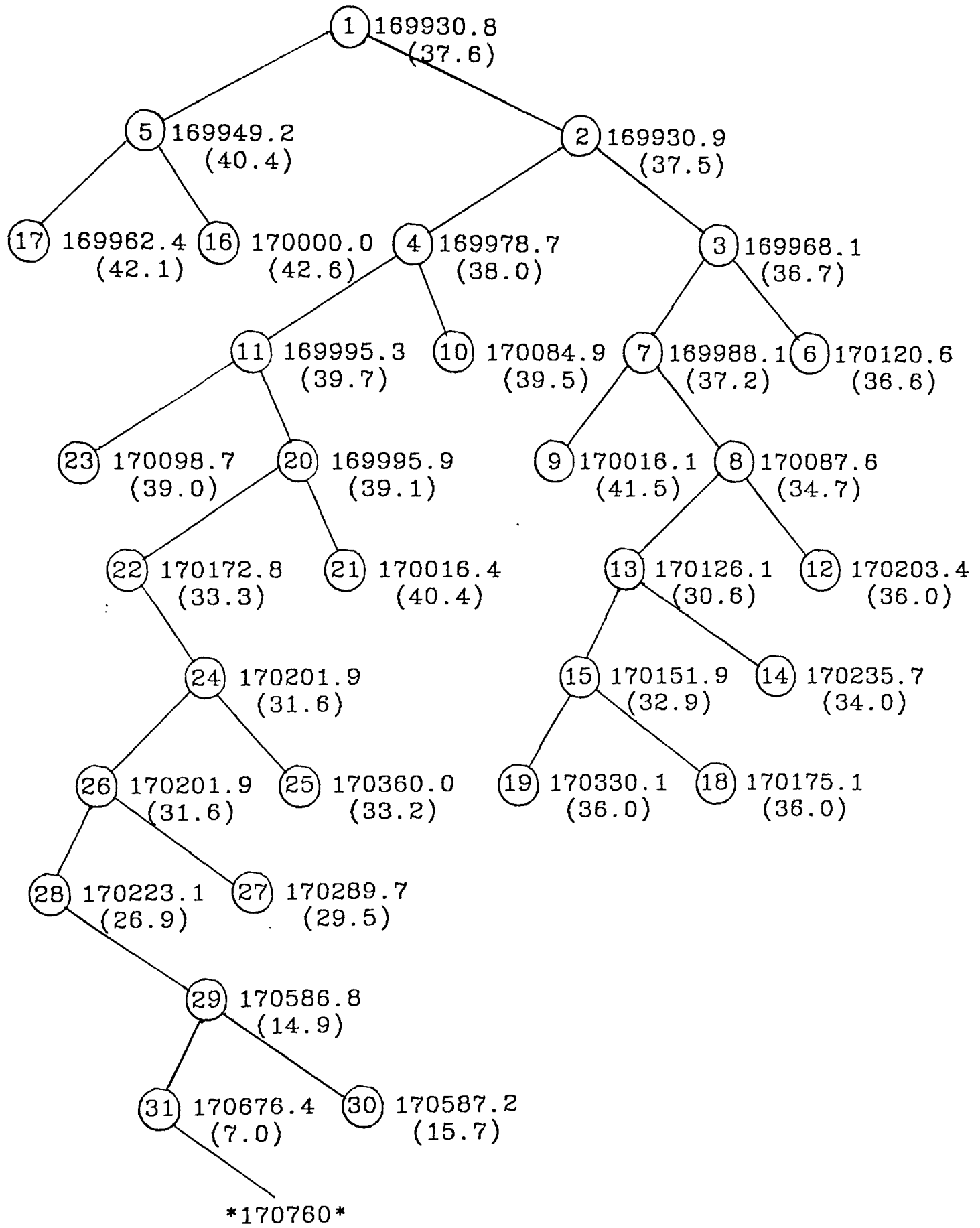


Figure 7.2. Example of constraint branching

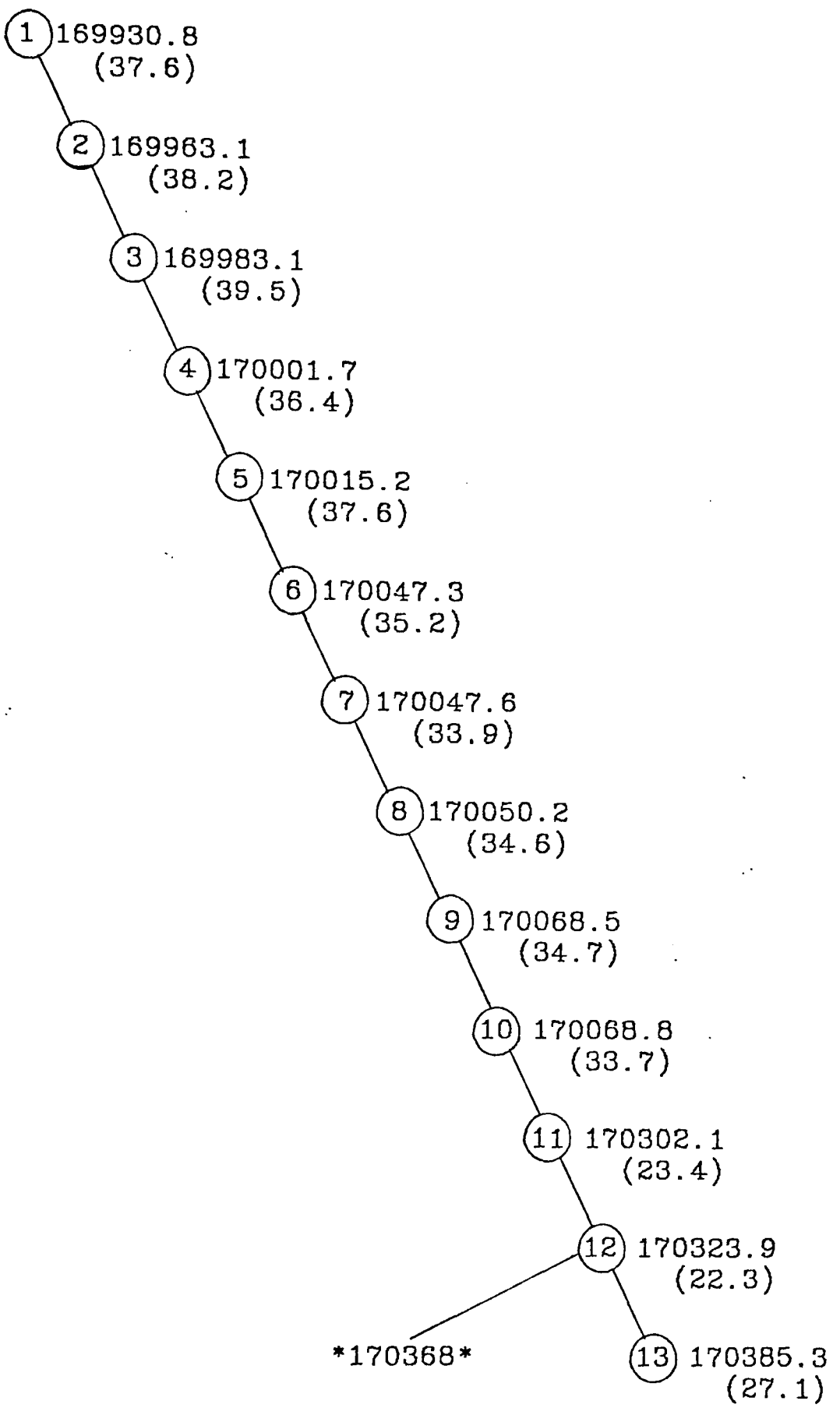


Figure 7.3. Example of relief time branching.

7.13 Comparison Of Branching Strategies

The results of using the constraint branching strategy and the relief time branching strategy on a representative sample of London Transport schedules are shown in Table 7.2. The problems are arranged in approximate order of difficulty, as measured by the time taken to find a good integer solution using constraint branching, except that the four problems for which the constraint branching strategy could not find an integer solution within 50 nodes are grouped at the end.

Table 7.2 shows that on some problems which are easy to solve using the constraint branching strategy, it can perform better than the relief time brachcing strategy. However, on the problems which are most difficult using constraint branching, the relief time branching strategy is much more successful. In particular, there is no problem in this sample for which no solution was found using relief time branching, although for the last problem shown in the table (HL207), the best integer solution is not sufficiently close to the continuous optimum to terminate the search immediately and the tree continues to grow until 50 nodes have been developed.

Since failure to find an integer solution is a vey annoying occurrence for schedulers using IMPACS, the fact that the relief time branching strategy does seem to reduce the incidence of failure is an important benefit. In cases where an integer solution can be found using constraint branching, the relief time branching strategy takes significantly less time, on average.

7.14 Failure To Find An Integer Solution

In spite of all the modifications to the branch-and-bound algorithm so that integer solutions can be found more quickly, it still occasionally happens that no integer solution can be found at all. Two different situations can occur:

a) the search terminates because all nodes of the tree have been fathomed through infeasibility;

b) no integer solution can be found by the time the tree contains 50 developed nodes.

In the first case, the infeasibility arises because of the side constraints operating during the branch-and-bound phase, i.e. that the target number of duties defined by the sum of the duty variables in the continuous L.P. solution should not be exceeded, and that no work should be left uncovered. This type of failure indicates that even though the bus

Problem	No. of duties in solution	L.P. optimum	Constraint branching *Best integer found*	*Time (sec.)*	*Nodes in tree*	Relief time branchin *Best integer found*	*Time (sec.)*	*Node in tree*
PM12 (M-F)	34	88465.4	88717	0.3	1	88629	2.6	3
S72 (Sun)	19	51350.0	51439	0.4	2	51491	0.9	6
TL47 (M-F)	39	98354.3	98452	1.1	5	98460	1.2	4
CA37 (Sat)	38	102561.0	102605	1.5	2	102605	1.3	2
NB71 (Sat)	17	45364.6	45531	8.1	14	45482	16.4	25
V27 (Sun)	34	91732.2	91873	16.5	23	91926	7.2	10
GM11 (Sat)	27	73591.3	74305	17.2	9	74305	18.2	9
SW77A (M-F)	59	150270.1	150448	18.8	12	150426	7.0	7
AP86/b (M-F)	50	124889.4	125015	23.2	16	124961	8.1	8
TC130 (M-F)	23	58283.7	58587	23.5	30	58632	7.6	11
BW25 (M-F)	25	66180.6	68750	23.7	16	68750	16.4	14
WH58 (Sat)	28	74499.2	74671	24.4	20	74688	21.2	26
WH69 (Sat)	43	112726.7	113388	46.0	31	114504	17.9	15
CF24A (M-F)	52	137763.7	138273	62.3	30	138323	19.6	10
PM36B (M-F)	67	169930.8	170760	92.3	31	170368	28.6	13
AP86/a (M-F)	62	160061.5	160861	115.6	45	160757	51.1	26
AP86 (Sat)	44	120484.8	None	69.7	50	120623	4.0	4
TL108B (Sat)	19	51136.1	None	31.7	50	52503	22.6	23
U101 (Sat)	32	86773.8	None	85.8	50	87385	28.7	22
HL207 (Sun)	32	85877.3	None	87.5	50	86546	72.4	50

Table 7.2 Comparison of constraint branching and relief time branching

schedule can be covered by the target number of fractional duties, any integer solution based on the current set of duty variables is likely to need an extra duty; because of the reduction technique described in section 7.3, a feasible integer solution may exist which has been excluded, but one would expect that there are very few such solutions, if any.

The action to be taken depends on whether or not the scheduler expects to be able to find a schedule with the target number of duties. If not, a new continuous optimum, with the sum of the duty variables increased by one, can be found by reoptimising the current solution and the branch-and-bound phase can be re-entered.

If the target number of duties must be achieved, the original set of duty variables must be enlarged, to give the branch-and-bound algorithms more choice and hence improve the chances of finding a feasible integer solution. This can be done by relaxing the duty generation parameters, by rejecting fewer duties after formation, or, if appropriate, by selecting more relief times so that more duties can be formed.

In the second case, in which no integer solution can be found in 50 nodes, the failure does not indicate that no integer solution exists (see the last entry in Table 7.1). It is more likely that several integer solutions exist, but that their objective values are much higher than the continuous optimum, so that a node choice strategy based on objective values makes them hard to find.

It would be possible to increase the storage limits and allow the program to develop more than 50 nodes before terminating. However, this would increase the maximum branch-and-bound time, without a proportional increase in the chances of finding an integer solution. A tree with 50 developed nodes using the constraint branching or relief time branching strategies has many active nodes, many of which will look more or less equally promising as avenues for further exploration; as the tree gets larger, it becomes less clear where to look next for an integer solution. Paradoxically, in this situation it is often helpful to generate a more restricted set of duty variables; by reducing the number of feasible solutions, the chances of finding an integer solution may be increased.

CHAPTER 8. PRODUCING THE SCHEDULE

8.1 Heuristic Improvements

The final program in the IMPACS suite takes an integer solution produced by ZIP, in the form of a set of identifiers corresponding to those variables with value 1, and translates this into a set of duties. The improvement techniques described in this chapter are applied, and the list of duties is printed in a form meaningful to a scheduler. The printing program also contains an editing routine, described in section 8.8, so that the scheduler can make changes to the duties if necessary.

The improvement techniques are similar to the refining procedures in heuristic scheduling systems such as TRACS and RUCUS. There are three reasons why an IMPACS schedule might be improved in this way. First, the solution produced by ZIP is not guaranteed to be an optimal solution to the set covering problem; second, because the formulation does not include all possible legal duties, the heuristics may be able to improve the schedule by bringing in duties which were left out. Finally, some criteria to be taken into account when judging a schedule cannot easily be dealt with in the set covering formulation. For instance, there is a preference in London Transport schedules to have duties signing on and off in the same order, but since L.T. duties do not have paid mealbreaks, there is usually no difference in cost if duties sign off in the wrong order. Further, any change which makes a pair of duties more similar in spreadover or time on duty, would be considered beneficial, other things being equal, but again such a change would not usually affect the total cost of the schedule.

The routine was originally written for London Transport in order to deal with this last problem, and in fact, many of the improvements found do not result in cost savings, but just change the ordering of the duties or make them more evenly balanced.

8.2 SWAP routine

One of the improvement routines looks at pairs of duties and considers exchanging portions between duties. Similar routines exist in RUCUS and TRACS. However, when only two-bus duties are considered, the only possible exchanges are: a) exchange the first spells of the duties and b) exchange the first spell of the duty which starts later with the second spell of the other. If three-bus duties are to be considered, there are more possibilities: each portion to be exchanged can consist of either one or two spells, and the middle spells of three-bus duties can be exchanged. However, any exchange which would create a four-bus duty and a two-bus

duty from two three-bus duties is not allowed.

The exchange routine first determines whether the new duties following the exchange would be valid. If so, the exchange is considered beneficial if the total cost of the duties is reduced: the total cost consists of all the costs calculated by the WAGES routine, together with any penalty costs, if applicable. This means that the total operating cost of the schedule can sometimes be increased in order to improve the quality of the duties.

If the total cost is unaffected by the exchange, it is still considered beneficial if the total spreadover decreases, which may happen if the first portion of one duty is exchanged for the second portion of the other. Failing that, the exchange is made if the spreadovers of the two duties are made more even. When the first portions of the two duties are being exchanged, this last condition is equivalent to ensuring that the duties sign off in the same order that they sign on.

For London Transport schedules, if the exchange has an effect on the total number of split duties, this is an overriding consideration. An exchange is not made if an extra split duty is created, but it is made if it saves a split duty, regardless of the effect on the total cost.

Examples of improvements made by the exchange routine to London Transport schedules are shown below. Each duty is followed by its main characteristics, i.e. time on duty/spreadover.

```
INTERCHANGE SPELLS 1154-1355 & 1144-1345, CREATING DUTIES:
  266:0604-1035
  253:1144-1345 = 7:18/ 8:25

  263:0608-1007
  254:1154-1355 = 6:46/ 8:31
NO SAVING
```

The exchange is made in order to make the spreadovers more even, since they were previously 8:35 and 8:21, and the duties now sign off in the same order that they sign on.

```
INTERCHANGE SPELLS 1324-1525 & 1454-1656, CREATING DUTIES:
  256:0655-1135
  256:1454-1656 = 7:28/10:45

  259:0735-1205
  263:1324-1525 = 7:17/ 8:34
NO SAVING
```

The duties had spreadovers 9:14 and 9:05 before the exchange and so both were classed as split duties (spreadover > 8:40). The exchange has reduced the number of split duties. This takes priority over the fact that

the duties now sign off in the wrong order.

```
INTERCHANGE SPELLS 1411-1805 & 1421-1802, CREATING DUTIES:
  8:0655-1011
 25:1421-1802 = 7:19/11:29
 12:1411-1615
 34:1632-1805
  5:1857-2200 = 7:21/ 8:13
REDUCTION IN SPREADOVER ALLOWANCE=      3
```

The spreadovers were 11:32 and 8:03 before the exchange, so that the total spreadover has increased, but because duties with spreadover longer than 8:16 are paid an extra allowance, the total cost is reduced. (Note that the first "spell" in the exchange in fact consists of two spells.)

```
INTERCHANGE SPELLS 1237-1347 & 1257-1407, CREATING DUTIES:
 161:0621-0910
 161:1017-1347 = 6:52/ 7:57
 167:1107-1217
 162:1257-1407
 167:1547-1915 = 7:00/ 8:40
NO SAVING
```

The exchange of spells of two three-bus duties has made one of them into a two-bus duty, which reduces the total penalty cost, but not the wages cost.

8.3 MOVE routine

This routine examines in turn every relief time at which crews change over, and considers moving the changeover to the previous relief time or to the following one. Again, similar improvement techniques are used in heuristic systems.

As with the exchange routine, the new duties must be valid, and the change is normally beneficial if it reduces the total cost, or leaves the total cost unchanged and reduces the total spreadover, or has no effect on the total cost or spreadover but makes the spreadovers or times on duty more equal.

The MOVE routine does not usually succeed in making any changes unless there are pairs of relief times which are close together, because otherwise the new duties are unlikely to be valid. This is probably not the case in heuristic systems, where some very short duties may be formed, allowing room for considerable extension. In IMPACS, however, all the duties formed are fairly efficient and cannot accommodate very much extra work.

When the MOVE routine can make improvements, it may allow relief

times which were not originally selected to be brought into the schedule.

8.4 SWAP3 routine

This routine involves exchanging spells of work between three duties, in special circumstances: two of the spells are contiguous, and the third is split into two as a result of the exchange.

As an example, the three duties:

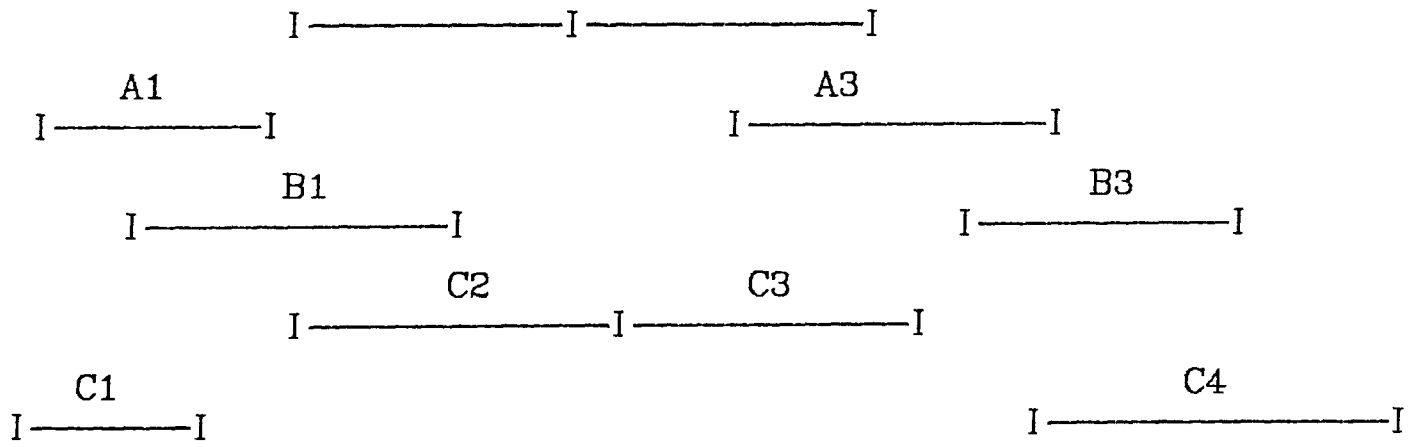
Bus 3: 1330-1740	Bus 4: 1600-2000	Bus 5: 1557-1812
Bus 1: 1908-2048	Bus 1: 2048-2317	Bus 2: 1930-2315

can be exchanged for the duties:

Bus 3: 1330-1740	Bus 4: 1600-2000	Bus 5: 1557-1812
Bus 2: 1930-2045	Bus 2: 2045-2315	Bus 1: 1908-2317

The two spells on Bus 1, from 1908 to 2317, are exchanged for the spell on Bus 2 from 1930 to 2315, which is split into two at 2045. The exchange would reduce the spreadover of the first duty by three minutes; the total spreadover of the other two is unchanged.

In general, three duties A, B and C are involved: a spell of A is contiguous with a spell of B and the two spells are combined into one as a result of the exchange, while a spell of duty C is split into two.



The current duties are A1+A2+A3, B1+B2+B3, C1+C2+C3+C4: the new duties will be A1+C2+A3, B1+C3+B3, C1+A2+B2+C4. Each of the spells A1, A3, B1, B3, C1 and C4 can consist of 0, 1 or 2 spells, so long as duties A, B and C each have a total of 2 or 3 spells, both before and after the exchange.

In the example above, spells A3, B3 and C4 do not exist, and in such a case the total spreadover is reduced whenever C2 (1930-2045) ends before A2 (1908-2048), provided that the signing off allowances are equal.

Another example is:

```
INTERCHANGE SPELLS 1123-1556, 1103-1326, 1326-1536,
CREATING DUTIES:
  223:0628-1006
  226:1123-1333 = 6:10/ 7:27
  226:1333-1556
  223:1636-2106 = 7:30/ 8:10
  228:0800-1016
  227:1103-1536 = 7:11/ 7:58
REDUCTION IN SPREADOVER ALLOWANCE=      3
```

Here, spells A3, B1 and C4 do not exist. If spell B2 finishes earlier than C3, the exchange will reduce the spreadover of the third duty, while leaving the total spreadover of the other two unchanged. In this example, the exchange has evened out the spreadovers of the first two duties (previously 7:20 and 8:17), which reduces the cost since duties longer than 8:16 are paid an extra allowance. The spreadover of the third duty has been reduced from 8:18 to 7:58, which also saves on spreadover allowance.

The criteria for deciding whether a valid exchange would be beneficial are the same as in the SWAP and MOVE routines.

This kind of exchange is valuable since it allows a much more radical change in the way the buses are broken into spells of work than the MOVE routine, which can only make marginal changes. The SWAP3 routine finds beneficial changes surprisingly often, and may allow further exchanges to be made using the SWAP routine.

8.5 Iteration

The improvement process is iterative: first all possible exchanges using the SWAP routine are considered, then all possible moves to adjacent relief times and then all possible exchanges using the SWAP3 routine, and this is repeated until no further improvements can be found.

8.6 Heuristic Improvements - Conclusions

Sometimes, the improvement routines can make significant cost savings and improve the quality of the schedule. They can also help to achieve objectives which cannot be expressed in mathematical programming terms. However, if the schedule is unsatisfactory it is often impossible to make any improvements using the heuristics, because extensive changes are required which are outside their scope. Moreover, the heuristics can only at best help to achieve the subsidiary objectives of avoiding undesirable duties and minimising the operating cost; they

cannot reduce the number of duties required.

In short, although the heuristics are useful, they are only a minor part of the total system. This contrasts with the experience of heuristic scheduling systems in which an initial solution is refined by improvement techniques similar to the ones described in this chapter. Indeed it was at one time thought by those developing such systems that the improvement techniques were sufficiently powerful to produce a good final result from even a poor quality initial schedule. Although this subsequently proved to be untrue, the improvement routines still play a major role in the creation of the final schedule, and it would be extremely unusual for them to be able to make no change at all to the initial schedule, as often happens in IMPACS.

The difference in experience is due to the totally different characters of the schedule produced by IMPACS and the initial schedule produced by a heuristic system. In an IMPACS solution, all the duties fit together neatly and all the mutual adjustments required to do this have already been carried out by the compilation process. Any major improvement in the quality of the schedule requires a fundamental rewriting of virtually the whole schedule. On the other hand, because the initial solution in a heuristic system is built up sequentially, the duties formed last are constrained by the duties already selected, and any change in the earlier duties allows considerable changes to be made throughout the schedule.

8.7 Printing The Schedule

When all the heuristic routines have improved the solution as far as possible, the schedule is printed as a list of duties. The list can be produced in two forms so far. One is a standard format intended to present the information in an easily-comprehensible layout; the other has been designed for London Transport and imitates their existing documents.

Although the standard format can be used and understood by schedulers, almost certainly, any other bus company which adopted IMPACS would also eventually want a specially-designed form of duty list, using the terminology which the schedulers are used to, and providing the information relevant to them. Even the order of the duties in manually-produced documents varies from place to place: the standard format and the London Transport format are alike in having the duties arranged in order of signing on time, but some duty lists have alternate early and late duties, while others have the early duties in order of

signing on time and the late duties in order of signing off time.

The printing program also presents the schedule in a graphical form, showing the duties covering each bus working. Although this has been modified to some extent to be similar to a working document used by London Transport schedulers, it is also produced for other users. An example of the IMPACS graphical output is shown in Figure 8.1; the bus schedule is the example discussed in chapter 4. Each spell of duty is labelled on the graph by a code, listing in order the buses which the duty works on. For instance, the three-bus duty "10,7,7" in Figure 8.1 works on bus 10, followed by two spells on bus 7. The code is not unique, as shown by the fact that there are two duties labelled "9,10", but it is usually possible for the scheduler to see which spells go together, and in any case the full details of each duty are given in the duty list.

8.8 Editing The Schedule

The scheduler may wish to make minor changes to the schedule, and an editing routine is provided for this, as part of the printing program. At present, editing is always necessary if the schedule has over-covered pieces of work, in order to produce a feasible schedule, since it is difficult to decide on standard rules for dealing with over-cover. If there is over-cover in the schedule, the duties involved are listed on the VDU screen, and the scheduler is then given the opportunity to edit the duties and remove the over-cover, by deleting each over-covered piece from all but one of the duties covering it, before the schedule is printed.

Apart from this, the scheduler may be able to improve the schedule manually, perhaps by exchanging one undesirable feature for another, in effect changing the priorities reflected in the penalty costs. In some cases, it may be possible to modify the bus schedule slightly in order to improve the crew schedule, and in others, some of the scheduling rules may be slightly flexible, so that the scheduler may be able to improve the schedule by forming duties which the duty generation program would reject as invalid.

The editing routine is interactive, and offers the scheduler a number of options, e.g. to exchange pieces of work between duties, or to take a piece of work from one duty and add it to another.

After printing the schedule, the scheduler can choose to store the list of duties in a stored schedule file, which can be picked up on a subsequent run of the printing program. The main benefit of this is that if the scheduler has made several alterations to the schedule, these will be maintained in the stored schedule file, so that if further changes are made later, the first batch need not be repeated.

LEEDS SAMPLE SCHEDULE

RELIEF POINT CODES: G = HEADINGLEY GARAGE
H = HEADINGLEY

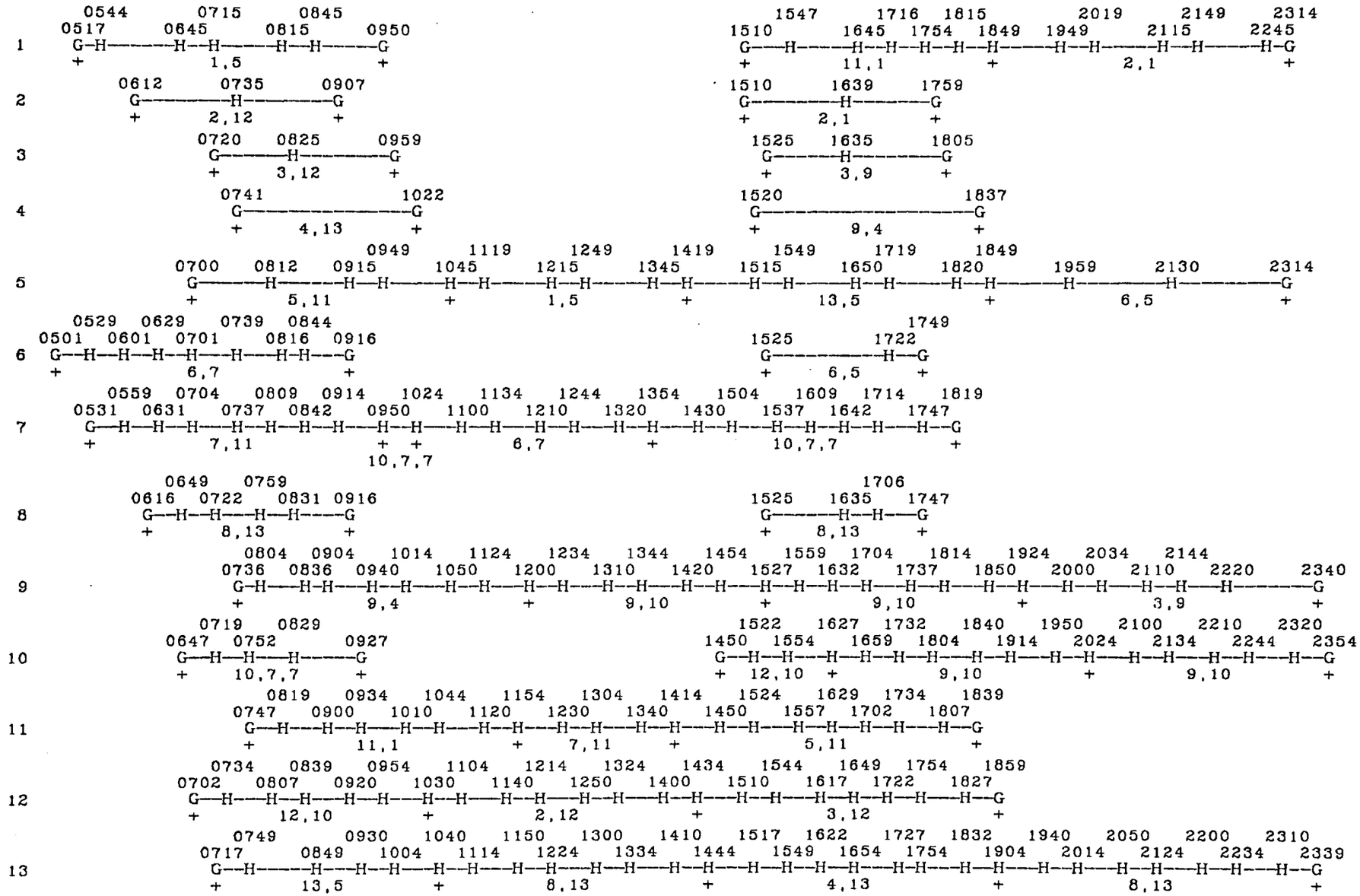


Figure 8.1 Bus Graph Output From IMPACS

CHAPTER 9. APPLICATIONS AND IMPLEMENTATION

9.1 Introduction

During the development of the IMPACS system, it was tested on many different scheduling agreements. This was important because the intention in developing IMPACS was to overcome a major drawback of heuristic systems, namely that they require considerable modification to work efficiently under new conditions. These applications to different agreements are described below in section 9.4.

9.2 Special-Purpose Routines

Although the intention in developing IMPACS was to make the system easily adaptable to new conditions, a certain amount of tailoring must be done for each agreement, to make sure that the duties generated are valid and are assigned the correct costs. At least two and possibly four special routines must be written for each new agreement, although this takes little effort. The routines required are described in this section.

9.2.1 VALID routine

The duty generation program must be able to test whether any potential duty is valid or not. The validation process is partly controlled by parameters whose values are defined by the scheduler via an input file, described in section 9.8. The rest of the validation involves a call to the VALID routine specific to the scheduling agreement, which checks the duty against any rules not covered by the parameters. The routine returns control to the calling program with an indication of whether or not the duty is valid.

In theory, it would be possible to extend the parameter file indefinitely, to cater for every agreement so far encountered, and to build more checks into the duty generation program, but since for every duty formed, many more are partly formed and then rejected, it would be very time-consuming to check every possible duty against parameters which may only be relevant to one agreement. Generally, clauses which have been found in several agreements are dealt with by parameters in the data file; anything which is peculiar to only one or two agreements is written into the VALID routine.

The VALID routine is often used to calculate the length of each stretch of a duty, and compare it with the maximum allowed in the agreement, since although agreements commonly specify a maximum stretch length, the allowances to be included in the length of a stretch

along with the driving time vary considerably. Other examples of the rules dealt with by a VALID routine are:

a) the second part of a split duty must sign on not more than ten hours after the first (Los Angeles);

b) in a Saturday schedule, if the duty has spreadover longer than 8:18, no spell of duty may exceed 4:30 in charge of the vehicle (London Transport).

Some simple agreements can be entirely dealt with by the parameters and the VALID routine consists merely of an immediate return to the calling program, whereas others can be quite complicated. However, even in complicated cases, the VALID routine does not usually take more than a day's work to write; the main difficulty is in establishing exactly what the agreement involves.

9.2.2 WAGES routine

As each duty is formed, the duty generation program assigns a cost to it by calling the WAGES routine. The cost, calculated in minutes paid, is passed to the ZIP program to form part of the objective contribution of the variable corresponding to that duty. It is clearly important that the costs should be calculated accurately, so that the ZIP program can minimise the true cost of the schedule.

It is also important from the scheduler's point of view that the costs shown on the final schedule listing produced by IMPACS should be reliably accurate, since otherwise the output will have to be checked by hand, if not rewritten, and users justifiably expect that a computer system should be able to relieve them of this kind of clerical task, whatever else it can do.

An immediate reason for calculating the cost of a duty in the duty generation program is that the agreement often specifies a maximum cost (and sometimes a minimum cost) so that the cost calculation is part of the validation process.

The cost calculation is done entirely by the WAGES routine, which can calculate a basic cost and up to two additional allowances. It is common for duties with a spreadover longer than a specified limit to be paid an additional allowance: for instance, this happens in Greater Manchester Transport schedules (spreadover > 9:04) and in London Transport schedules (spreadover > 8:16). There may also be a daily overtime payment if the basic cost exceeds a certain amount, or an unsocial hours allowance if the crew is on duty before or after specified times. The maximum and minimum cost parameters, and the minimum

paid day, if any, apply only to the basic cost of the duty.

9.2.3 Overtime And Penalty Cost Routines

Whereas the VALID and WAGES routines must be written for each new agreement, the other two special-purpose routines may not always be required. One adds possible overtime pieces, or one-bus duties in general, to the list of duties formed by the main duty generation program, as described in section 5.7.

The other is called by the ZIP program to assign penalty costs to some kinds of duty, if the scheduler so specifies. As described in section 6.5, the penalty costs form part of the objective contribution of the duty, and are designed to discourage the use of these kinds of duty. The descriptions of scheduling rules often contain statements such as: "There is reluctance on the part of Union Representatives to accept middle duties finishing after 2000 hours" (GMT Bolton schedule), or, "The number of duties with a break longer than 60 minutes should be minimised" (Amsterdam), so that it is usual to require penalty costs, but there is no similarity between agreements as to the kind of duties which are to be avoided, so that it would be impossible to devise any set of parameters which could replace a penalty cost routine.

9.3 Data Files

Four data files are required to run the IMPACS system. The contents of each file are described in detail in the Appendix.

The first file contains the relevant bus schedule information, which consists mainly of the list of relief opportunities on each running board. Each relief opportunity is defined by a relief point and normally a single relief time; however, IMPACS stores two separate relief times, one applicable to the crew leaving the bus and one for the relieving crew. These are in most cases identical, but in situations where the bus can be left unattended, for instance at bus stations, and the crew being relieved can leave the bus before the next crew starts, two different relief times can be specified. The file also lists the relief points and gives the signing on and off allowances at each relief point and, for every pair of relief points, the minimum mealbreak allowances and so on.

Except for very small schedules, this is the largest of the four files, and it can be a time-consuming task to extract the lists of relief opportunities from the bus schedule and set up the file. However, if the bus schedule has itself been compiled by computer, the relief information can be compiled automatically. London Transport have a

program to link their own bus scheduling system with IMPACS in this way, and it is likely that future users of IMPACS will also be users of Wootton Jeffreys' TASC bus scheduling system, which can produce the relief information in the required form already. Greater Manchester Transport, who have been using IMPACS for some months, use the VAMPIRES system for bus scheduling, but have not yet adapted it to produce the relief information automatically.

The second file contains the duty generation parameters, and is required when forming and validating possible duties, as described in chapter 5.

The ZIP program requires a small data file controlling the way in which it is to be run. As described in section 6.9, the program can be run in several different modes, for instance allowing the scheduler to pick up an existing L.P. solution, vary the conditions and reoptimise. The mode to be used is specified in the ZIP data file, and any side constraints are listed at the end of the file. The scheduler also specifies in the file the constant element of the objective contribution of a duty variable, and the parameters used in calculating the costs of slack and surplus variables, as described in sections 6.5 and 6.6.

The fourth data file is is used by the relief time selection program. It specifies various parameters to be used by the program, the most important being the spell lengths to be used in marking time-bands on the running boards, as described in chapter 4.

To keep the IMPACS system independent of hardware and operating system, no special editor has been provided to help in setting up these files. However, where it would otherwise be difficult for the scheduler to remember the significance of the data items required, the reading routines have been written so that each line of data in the file is preceded by a line of text. The text line can be used by the scheduler to describe the items on the next line and is ignored by the reading routine. A skeleton data file can be created, consisting of the lines of text alternating with blank lines; whenever a data file is to be set up, the skeleton can be copied and the blank lines filled in, using the standard system editor provided. This minimises the possibility of error and makes setting up the data quick and simple, although it does not provide on-line validation and prompts as a special-purpose editor could do.

9.4 Applications Of IMPACS

9.4.1 Leeds Schedules

Much of the early development work on IMPACS was done on schedules from Leeds District of West Yorkshire PTE. The manual schedules were not available, but the schedules had previously been compiled for West Yorkshire using TRACS. Since West Yorkshire had used TRACS extensively and the system had been carefully tailored to meet their scheduling conditions, the quality of the schedules produced was very high, and would be at least as good as the manual schedules.

Two Monday to Friday schedules were used, requiring 19 and 72 duties; the larger schedule was also split up into two parts, by bus route, to give schedules requiring 31 and 44 duties. In this way, four schedules of different sizes, all with the same union agreement, were obtained.

Most of the programs in the IMPACS system were initially developed using these schedules, and because this early work was inevitably influenced by the specific requirements of these schedules, eventually excellent results were obtained.

9.4.2 Leicester

One of the Monday to Friday schedules operated by Leicester City Transport was compiled in 1981 purely as a test exercise for IMPACS, since the data had been collected for another purpose. The union agreement was simple and allowed very long duties: the maximum spreadover was 9:45 for straight duties and 12:30 for split duties, and the average platform time in the current schedule was 7:52, which is unusually high. Possibly in compensation for the longer than average hours, the split duties were paid through.

Because of the long hours worked, the pattern of coverage of the bus schedule was very simple. There were no reliefs during the peaks, all the straight duties covered one peak and all the split duties covered both peaks. It could be shown that unless the number of split duties was increased (which was assumed to be forbidden), the schedule had the minimum possible number of duties (59).

The relief times were only 44 minutes apart on average, which could have given rise to a very large number of constraints, but because it was clear that none of the relief times during the peaks should be used, these times were banned when the relief time selection program was run, and this difficulty did not arise. An IMPACS schedule with 59 duties was produced, and appeared to be slightly cheaper than the current schedule.

9.4.3 Cleveland Transit

The data for two Cleveland Transit depots (North and South) was collected during an exercise several years earlier involving TRACS. The scheduling conditions at the two depots were sufficiently different to be counted as separate agreements, although the main feature of both was that split duties were not allowed, even in Monday to Friday schedules.

At North Depot, the maximum spreadover allowed was 9:10, but only a limited number of duties could have spreadover longer than 8:25. In order to solve the problem of covering a peaked bus schedule without using duties with long spreadovers, some of the early duties had overtime work attached on "evening specials" (short running boards over the evening peak). Before compiling the IMPACS schedule, the work covered by these attached evening specials was removed, because it did not appear that there were any restrictions on the early duties involved, and it was not clear how the overtime work was paid. The Monday to Friday manual schedules also left some morning peak work uncovered.

All three of the two-person-operated (TPO) schedules were compiled using IMPACS - Monday to Friday, Saturday and Sunday. In each case the IMPACS schedule was cheaper than the manual schedule and had fewer long spreadover duties, and the Saturday IMPACS schedule had two fewer duties. The Monday to Friday one-person-operated (OPO) schedule was also compiled and had two fewer unallocated pieces of work than the manual schedule, and again fewer duties with long spreadovers.

For the South Depot, only the OPO schedules were considered, because the TPO schedules were very small. The South Depot duties were even shorter than at North Depot, since although the maximum spreadover was 9:00, only a limited number of duties could be longer than 8:30, and the average spreadover had to be at most 8:00. The duties in the manual schedules in fact had a very low work content, averaging about 5:35 in each case.

In order to limit the number of duties with spreadover longer than 8:30, and to reduce the average spreadover to less than 8 hours, the facility for adding penalty costs to certain kinds of duty was introduced into the ZIP program. All duties with spreadover longer than 8 hours were penalised, and a higher penalty was attached to duties longer than 8:30.

The IMPACS schedules had either fewer duties or fewer unallocated pieces than the corresponding manual schedules, saving six unallocated pieces in the Monday to Friday schedule, two full duties in the Saturday

schedule and one on Sunday. The number of duties with spreadover greater than 8:30 was also reduced, except on Sunday when the IMPACS schedule had the same number as the manual schedule.

Compiling the schedules took little computer time and integer solutions were found quickly; in fact, the Saturday schedule gave an integer L.P. solution.

9.4.4 Cardiff

This schedule was for Monday to Friday. The Cardiff agreement limited the proportion of split duties to 10%, which made it difficult to cover the morning and evening peaks, although the peak:off-peak ratio was relatively low in this case (1.4). Many evening peak buses were covered by overtime in the manual schedule, and the agreement stated that overtime should be attached to approximately one in seven duties. There were also limitations on the number of straight duties with spreadover more than 8 hours.

The overtime pieces appeared in the IMPACS schedule automatically as slack variables on evening peak pieces of work. To produce schedules equivalent to the manual schedules, a constraint was put on the number of split duties, and penalty costs were attached to duties with spreadover longer than 8 hours. This automatically produced a schedule with the right proportion of overtime work. The manual schedule had 81 duties and 12 overtime pieces; the IMPACS schedule had two fewer overtime pieces.

The IMPACS schedules were shown to Cardiff scheduling staff, who said that they were impressed by the schedule and that the duties were very similar in style to those they would form themselves. However, the work from this crew schedule had recently been integrated with the other Cardiff schedules, so that there was no question of implementing the IMPACS schedule, and the new integrated schedule was at that time felt to be too big to solve.

9.4.5 West Midlands Selly Oak Garage

As part of another project, information had been collected from West Midlands Passenger Transport Executive on their crew scheduling practices, and this included data for the Monday to Friday and Saturday schedules relating to a group of bus routes operated by Selly Oak garage. These two schedules were used as a test exercise for IMPACS at the end of 1981.

One of the rules of the union agreement specified a maximum

average cost for the duties in a schedule of 7:48 for Monday to Friday schedules, and 8 hours for Saturday schedules. For the two Selly Oak schedules, this rule turned out to be critical: without it, fewer duties would have been required to cover the work. The calculation of the cost of a duty was extremely complicated: for instance, mealbreaks were paid in duties signing on before 0545, late duties and duties with a mealbreak of less than an hour, and there was a make-up allowance for short spells of duty which could be set off against the paid mealbreak time, if any. Hence it was difficult to estimate the average cost of the duties in advance, and it must have been a difficult rule for manual schedulers to work to.

It would have been possible to introduce a constraint on the average cost of a duty into ZIP, but this was not done, partly because a maximum average cost rule is not common, and did not warrant a general modification to the program, and partly because it was not initially clear whether the constraint would be critical. Instead the rule was dealt with by forcing more duties into the schedule, to reduce the average cost.

ZIP had already been modified to allow constraints on the number of duties of any type (earlies, lates, etc.) to be specified initially. The first Selly Oak schedules, however, were compiled without any constraints on the number of duties, so that the work was covered in the minimum number of duties, regardless of the average cost. Because the average proved to be too high, the ZIP program was further modified to allow constraints on the number of duties of any type to be added after an initial solution had been found, and the solution reoptimised. Extra duties were gradually added until the average cost of the schedules was sufficiently low.

The simplest way of doing this would have been to increase the total number of duties, but at that time only constraints on the individual types of duty had been incorporated into ZIP, so that lower limits on the number of each type of duty had to be used in order to increase the total number of duties. This was a tedious process, particularly as several different duty mixes can give the same total.

For the Monday to Friday schedule, the unconstrained solution, with the minimum number of duties, had 71 duties with an average cost of 8:13. Adding more duties to reduce the average to 7:48 gave a schedule with 75 duties, compared with 78 duties and two unallocated pieces of work in the manual schedule. For the Saturday schedule, the minimum number of duties was 62, and a schedule with a legal average duty cost was found with 64 duties, compared with 66 duties and two unallocated pieces in the manual schedule.

These schedules were discussed with West Midlands schedulers, who were "generally quite impressed", but some minor modifications to the construction of duties were required. The Monday to Friday schedule was recompiled following these changes and then required an extra duty, although there was still a considerable saving compared with the manual schedule. A schedule for another depot was also successfully compiled, at the schedulers' request, but they did not proceed with further investigation of IMPACS, because they felt that an interactive amendment facility was essential. At that time IMPACS did not have an editing facility, and they felt that the Wootton Jeffreys' COMPACS system might be more suitable for their requirements.

Because of the cumbersome way in which duties had to be added to reduce the average cost, the ZIP program took much longer to run than it would with the present version of IMPACS.

9.4.6 Amsterdam Route 15

As described in section 2.3.2, a crew scheduling system using a mathematical programming approach has been tested in Amsterdam on the set of schedules for one bus route: the tests were described at the Leeds Workshop in 1980 [21]. After discussions with Roes, one of the authors, he sent the data for the Saturday schedule in 1982, and this was used as a test schedule for IMPACS. The result was discussed with staff of the Amsterdam bus company, who agreed that it would be acceptable and later sent the data for the Monday to Friday schedule; unfortunately, no comments on the IMPACS schedule have been received.

The most notable feature of these two schedules was that the late duties were single spells of work of up to 8:45 in length. In fact, the bus schedule had no relief times shown during the evening, and all the late duties were determined in advance. The facility for prespecifying duties was introduced into IMPACS as part of this exercise to cope with the late duties. Prespecifying the late duties left, in effect, a truncated bus schedule finishing at about 1800, just after the evening peak.

An unusual feature of the union agreement was that the distinction between straight and split duties was based on the length of the mealbreak rather than the spreadover. Straight duties had a mealbreak of 1:15 or less; split duties had a mealbreak of at least 1:45. The maximum spreadovers were 10 hours and 12 hours respectively and a split duty could have a shorter spreadover than a straight duty. Split duties were allowed in both the Monday to Friday and Saturday schedules, but their number was to be minimised.

One difficulty with the Saturday schedule was that although the number of three-bus duties was to be minimised, those required were of a type not previously formed by the duty generation program, which had to be modified accordingly; these modifications were also required for the Monday to Friday schedule. After these changes, the Saturday schedule was very easy and quick to compile. Because the 13 late duties out of the total of 32 duties were prespecified, the problem was actually very small. Leaving aside the work covered by the late duties, the relief times occurred only every 2 hours on average, in both schedules, and the relief time selection procedure was not used. The set covering problem for the Saturday schedule had only 74 constraints and 1040 variables, and took only a few seconds to solve. The IMPACS schedule was of comparable quality to that produced by the DIGOS system: both had the same number of duties as the manual schedule but one fewer split duty.

The Monday to Friday schedule had 65 duties of which 15 were the prespecified late duties. The manual schedule which was sent with the data had eight split duties, and since the number of split duties was to be minimised, this was taken as the maximum allowed. An IMPACS schedule with 65 duties and eight splits was produced, together with an alternative schedule with 63 duties and ten splits. In most cases, the second schedule would have been preferable, since the requirement to minimise the number of duties would have higher priority than minimising the number of split duties. These two schedules were found in a total of about 170 seconds (on an Amdahl 470/V7).

9.4.7 East Kent Road Car Company

At the end of 1982, trial schedules were compiled for two National Bus Company subsidiaries, at the request of NBC Computer Services, who were investigating computer scheduling systems. One of the test schedules was the Monday to Friday schedule for the Canterbury depot of the East Kent Road Car Company, the other was a Yorkshire Traction schedule described in the next section.

Both of these schedules covered services operating mainly from a central bus station. The bus could be left unattended at the bus station between the arrival time and the departure time, if the crews were required to change over there. This required a change to the system, because it had previously been assumed that the relief time was the same for the crew being relieved and for the relief crew.

Considerable difficulties were experienced in compiling both schedules because the manual schedule could not be examined until the

end of the exercise. This was understandable, because NBC wanted to ensure that the schedules produced were not influenced by knowledge of the existing duties. However, it made it difficult to establish exactly what should be allowed, because the scheduling staff tended to describe what should ideally happen, rather than the rules actually followed.

The Canterbury schedule was required to have not more than 58 duties and 12 overtime pieces, with the same proportions of the different types of duty as the manual schedule. Split duties could only have a maximum of 10:30 spreadover, but as the evening peak was unusually early (about 1530 to 1630), some of the split duties could cover both peaks.

A first schedule was sent within a week, but was unacceptable because the overtime pieces were too long, due to a misunderstanding of the rules. The Traffic Manager also commented that several of the duties were too short or had too long a mealbreak, which caused some problems with over-cover in the second set of schedules, but even so with the overtime pieces reduced in length they were accepted as satisfactory. Two schedules were sent: one with the same number of each type of duty as the manual schedule, and the other with 56 duties and 14 overtime pieces.

When the manual schedule was subsequently made available, it was found that several of the duties were as short as the ones which had been objected to in the IMPACS schedule. This illustrates the difficulty in establishing exactly what the scheduling rules are: although it may be a useful guideline for schedulers to keep duties above a minimum length if possible, to achieve an efficient schedule, it is rare for a duty to be unacceptable simply because it is too short, but schedulers may include guidelines such as this when specifying the rules.

Apart from the problems in interpreting the rules, there were no difficulties in compiling the schedules: each run took between 25 and 35 seconds approximately and either the L.P. solutions were naturally integer, or an integer solution could be found quickly.

9.4.8 Yorkshire Traction

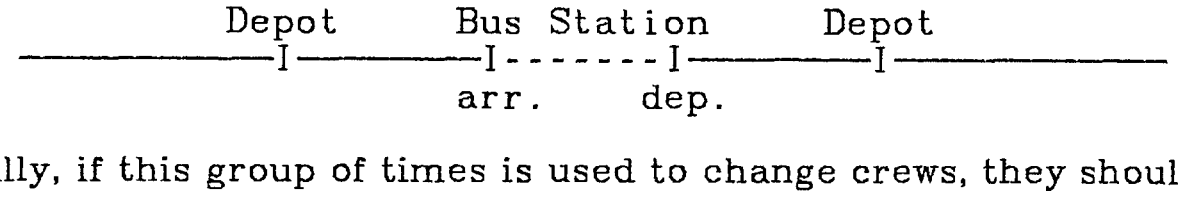
The trial schedule for Yorkshire Traction was the Monday to Friday schedule for the Huddersfield depot, which required a maximum of 58 duties, with no split duties.

Again, there was considerable difficulty in establishing the scheduling rules. For instance, it was originally stated that the maximum spreadover should be 9 hours; after failures to compile a 58-duty

schedule with this rule, it was proved to be impossible and subsequent enquiry revealed that the maximum spreadover occuring in the manual schedule was 10:20. When the manual schedule was eventually examined, it was found that many of the duties broke the rules used for the IMPACS schedules; nine of them had "illegal" mealbreaks, for example. In spite of these problems, a 58-duty schedule was produced within two weeks and was accepted with enthusiasm by Yorkshire Traction. They commented that it was of suitable quality for immediate issue to the depot, and appeared to be slightly cheaper than the current schedule.

An interesting feature of the schedule demonstrated again that set covering is a much better model than set partitioning for some crew scheduling problems. Many of the routes operated by the Huddersfield depot are very long and terminate at the bus station, passing the depot on the way; it is ten minutes by bus from the depot to the bus station. There is no canteen at the depot, so that all mealbreaks are taken at the bus station, whereas crews must sign on and off at the depot.

Because of the route structure, relief times occur in groups, as shown:



Ideally, if this group of times is used to change crews, they should either change over at the bus station, when one crew is just starting a mealbreak and the other crew is just finishing a mealbreak, or one crew should be signing on and the other signing off, and the changeover should take place at one of the relief times at the depot.

However, this cannot always be arranged. If one crew is signing on and the other is about to take a mealbreak, the changeover can take place either at the first depot relief time or at the bus station, with one crew travelling on the bus as passengers. Since the travelling time counts as working time, it makes no difference which crew are the passengers. The piece of work from the depot to the bus station is in effect assigned to two crews, which can be represented as over-cover.

Because the set covering formulation allows over-cover, the duty generation program need only form duties which start a mealbreak at the bus station, handing over either to a duty just finishing a mealbreak or to a duty which has previously signed on at the depot. If a set partitioning formulation were being used, two duties would have to be formed, one starting a mealbreak at the depot and the other at the bus station.

Similarly, with set covering, duties need only finish a mealbreak at

the bus station, and if taking over from a duty signing off, one of the two crews would travel as passengers to the depot.

Because there were long intervals between these groups of relief times, the relief time selection procedure was not used for this schedule.

9.4.9 Yorkshire Traction Wombwell Depot

Following the Huddersfield depot exercise, NBC paid for an IMPACS schedule to be compiled for Yorkshire Traction's Wombwell depot at the end of 1983.

The main complication in this schedule was that the depot operated many bus routes which did not run through Wombwell, so that there were many different relief points and the driver had to travel between the depot and the relief point by catching another bus. Fortunately, in most cases when this happened the frequencies of the route being operated and the route used by the driver to travel on were such that the total travel time was constant for much of the day, so that the problem could be dealt with by using fixed travel times, with duplicate relief points to allow for the variations.

For this exercise, the current crew schedule was sent with the rest of the data, which was valuable because it showed how various unwritten rules should be treated. For instance, it was stated that although the minimum mealbreak (net of travel time) was 30 minutes, breaks should preferably be longer than this. The manual schedule had some duties with a mealbreak of only 30 minutes, so this rule was dealt with by adding penalty costs to any duty with a minimum mealbreak of less than 40 minutes.

In this schedule, no split duties were allowed, and the maximum spreadover was 9:30.

There were originally 351 pieces of work in the bus schedule, which were reduced to 221 by the relief time selection procedure. The ZIP program then took just under 300 seconds to run (on an Amdahl 470/V7).

The current schedule had 80 duties, and the IMPACS schedule, covering exactly the same bus schedule, had 72 duties. This schedule was accepted by Yorkshire Traction, and in fact implemented, giving an annual saving of £50,000.

9.4.10 Yorkshire Traction Rawmarsh Depot

The third schedule which was compiled for Yorkshire Traction, at the end of 1984, was in some ways the most complicated and difficult of

the three.

The schedule was operated by five different rotas, each of which had different rules concerning the the bus routes which could or could not be worked. Unfortunately, every route could be worked by more than one rota, and there was a considerable number of schools and works contract buses which could be worked by any rota, so that the schedule could not be compiled in smaller units. Most of the VALID routine in this case was concerned with ensuring that the duties formed could be operated by at least one rota. In addition, the proportion of duties to be worked by each rota was fixed, so as an ad hoc measure, the duty type classification was used to represent the different rotas, rather than early duties, late duties, etc., so that limits could be set on the number of duties in each rota.

A maximum of eight split duties were allowed: the maximum spreadover for all other duties was 8:30. The maximum continuous driving time was 4 hours, but Yorkshire Traction specified that anything longer than 3:30 should be avoided, if possible.

It was extremely difficult to find integer solutions in this exercise. This was probably due to the high proportion of three-bus duties, both in the set of duties forming the variables of the set covering problem and in the final schedule. There were many very short bus workings, and in addition, the restriction to less than 3:30 continuous driving time whenever possible meant that, because of the intervals between relief times, on many buses spells of duty had to be much shorter than 3:30. The combination of these two factors meant that in many cases it was only possible to form efficient duties by working on three buses. The final schedule in fact had 36 three-bus duties out of a total of 75.

The final run of the ZIP program took 970 seconds of computer time (on an Amdahl 470/V7); because of an initial misunderstanding of the scheduling rules and the difficulty in finding an integer solution, several runs were necessary. However, the current manual schedule for Rawmarsh depot had 81 duties, and the exercise produced a 75-duty schedule which Yorkshire Traction accepted.

The differences between the three Yorkshire Traction schedules show the amount of effort that may be required to implement a computer scheduling system for a bus company when there is no common set of scheduling rules.

Some of the differences could be dealt with by changing parameter values, for instance the treatment of split duties. Even though all three

were Monday to Friday schedules, the rules concerning split duties were quite different. The Wombwell and Huddersfield schedules supposedly had no split duties, although the maximum spreadover allowed in practice at the Huddersfield depot was 10:20, which would normally be counted as a split duty spreadover. The Rawmarsh depot had a limited number of split duties with up to 11:30 spreadover.

There were also considerable differences in the way duties were costed at the three depots. In the Huddersfield schedule all the duties were paid through, and this was also true of the Rawmarsh straight duties, whereas the split duties had an unpaid break. At Wombwell depot, however, the rule was more complicated: duties with spreadover up to 8:18 were paid through, but in longer duties up to 45 minutes of the mealbreak was unpaid. It is difficult to see how these variations could be dealt with except by tailoring the WAGES routine to fit the conditions of each depot in turn. Moreover, it is quite possible for Saturday and Sunday duties to be costed differently from weekday duties at the same depot.

Essentially, the three depots had to be treated completely independently, with separate VALID, WAGES and penalty cost routines (if appropriate) for each, and probably if work were done for other Yorkshire Traction depots, new routines would be required in each case. However, it should be noted that even starting from scratch for each exercise, the three schedules were produced very quickly: it would probably have taken a manual scheduler at least as long to compile schedules of this size.

9.4.11 Greater Manchester Transport Bolton Depot

A schedule was compiled as a demonstration of IMPACS to Greater Manchester Transport in May and June, 1983. The schedule which they selected was a Monday to Friday schedule for Bolton depot, and was small by GMT standards, since many GMT schedules have more than 100 duties.

The current manual schedule had 37 duties and eight short pieces of overtime. However, it was felt that this was not a very efficient schedule and it was recompiled by one of the more expert GMT schedulers, although ignoring some of the local preferences which had been taken account of by the Bolton scheduler. The new schedule had 36 duties and one overtime piece and was used as the basis for comparison with the IMPACS schedule. It is debatable whether this is a sensible way of judging computer scheduling systems, since if they can produce good results compared with average schedulers they may be worthwhile, even if they can be beaten by expert schedulers.

Several of the duties in the new manual schedule had a very short

joinup during the evening peak. These duties were shown as three- or (more usually) four-bus duties, but in fact, pairs of duties were involved, where the buses were at the same relief point at the same time, and if the schedule had been implemented, the bus schedule would have been modified by relinking the buses to follow the crews in these cases. The spells either side of the joinup would then have appeared as a continuous spell of work on one bus.

Constructing duties in this way is quite often done in GMT schedules, to ensure that the late duties are as efficient as possible. It would be very difficult to produce duties of this type using IMPACS, partly because of the problems which would arise if four-bus duties were to be formed, and partly because a pair of duties are involved in the relinking, whereas IMPACS would not guarantee that both of the pair would appear in the solution.

The IMPACS schedule in fact had 36 duties and two overtime pieces and so was slightly worse than the manual schedule, although an earlier IMPACS schedule which followed the same rules as the Bolton scheduler had fewer duties than the manual schedule currently operating. For the second schedule, the set covering problem had 159 constraints, selected from 299 pieces of work originally, and 1767 variables, and was solved in 81 seconds (on an Amdahl 470/V7). The GMT comment was that in spite of the inability to relink buses, "overall, however, the results are very impressive" and they suggested that the schedules produced would be acceptable if an interactive facility were introduced into IMPACS, for instance by a link with the COMPACS system, to allow the scheduler to relink buses manually.

9.4.12 Los Angeles

A trial schedule was compiled in October 1983 for one of the operating districts of the Southern California Rapid Transit District (SCRTD) based on Los Angeles. The schedule was very large compared to earlier schedules which had been compiled using IMPACS: there were about 115 buses in each peak, falling to 62 in the middle of the day. The bus schedule contained work on many different routes, with varying frequencies of relief times, but each running board worked on a single route. The scheduling agreement had some features which are common in North America but do not occur in this country.

Three types of duty were allowed:

a) straight runs, working on one bus without a break. These had to be at least 7 hours long, with a maximum driving time of about 10:15.

b) split runs, consisting of two pieces of work separated by a mealbreak. Interlining, that is working on two different bus routes, was not allowed except in restricted circumstances. The mealbreak had to be between 34 minutes and 3 hours long, and the maximum spreadover was 13 hours. The restrictions on driving time were as for straight runs.

c) trippers, to be worked as overtime. These were either short peak-only running boards, or longer pieces over one of the peaks with a driving time of between 3 hours and 5:15, approximately.

Often when straight duties without a mealbreak are allowed, many of them can in effect be specified in advance, leaving an assortment of peak pieces to be fitted together to form split duties. However, in this case two short peak pieces could not be combined to form a valid split run (because the mealbreak would be too long), and for this reason it was impossible to assign any work to a straight run in advance, except on a few very early buses.

Because of the high peak to off-peak ratio, most of the split runs consisted of a peak-only board together with a longer piece of work either over the other peak or late in the evening. However, because of an additional rule, that the second piece of a split run must sign on not more than 10 hours after the first, it was not always possible to combine work on an early bus with an evening peak board.

It was necessary to reduce the number of relief times in the problem considerably. This was done manually, because the relief time selection program does not cater for one-bus duties. A few simple rules were devised: for instance, since trippers must be at least 3 hours long, unless covering a whole board, no relief time within 3 hours of the start or end of a board need be selected. Using these rules, the number of pieces of work was reduced by about half to 398, which is still considerably more than IMPACS can normally cope with.

The number of duties generated was small considering the number of pieces of work: about 2500 split runs and about 500 straight runs and trippers. A total of four different IMPACS runs were done, varying the conditions to achieve a balance of duties similar to that in the manual schedule. Each run took 20-25 seconds, and in every case the solutions to the relaxed L.P. were integer.

The final schedule produced appeared to contain duties similar in quality to the manual schedule and had three fewer trippers (87) when constrained to have the same total number of straight and split runs as the manual schedule (122). This may not be a fair comparison, since

without discussion with schedulers it is impossible to be certain that the agreement has been properly understood, although wherever the rules were rather hazily expressed, a strict interpretation was chosen, and the rules in the manual schedule often broke the rules adopted for IMPACS.

The most interesting aspect of the exercise, however, was that even though it was larger than any previous IMPACS problem, it could be solved quickly. Whereas previously the upper limit on the size of schedules which can be compiled by IMPACS had been thought to be about 80 duties, it now became clear that in the right conditions much larger schedules could be compiled without difficulty.

9.4.13 Melbourne Transit Authority

Two schedules from the Melbourne Transit Authority, Australia, were compiled in October, 1984. The Melbourne schedulers had been using the COMPACS system, described in section 2.2.6, for some time, but found that the automatic mode did not always achieve the same number of duties as the manual schedules. They sent a set of schedules to Wootton Jeffreys for further investigation. Since COMPACS is not expected to produce good schedules without assistance from the scheduler, it was felt that their requirements for an automatic scheduling system would be better met by IMPACS, and the author compiled two of the schedules using IMPACS as a demonstration.

The sample schedules chosen were the Sunday and weekday schedules for one depot; these were the schedules for which the discrepancy between the COMPACS and manual schedules was greatest. The Sunday schedule was very small (9 duties); it commonly happens with heuristic systems that very small schedules are difficult to compile successfully, because there is little flexibility. As expected, a 9-duty schedule was easily found using IMPACS; the ZIP program took 15 seconds to run (on an Amdahl 5860).

The weekday schedule required 41 duties, with a maximum of 15 split duties. The bus schedule had over 450 pieces of work, which was reduced by the relief time selection program to 164, so that the problem was then of a manageable size.

There was no difficulty in finding a 41-duty solution with 15 split duties, as required; the solution appeared to be slightly cheaper than the manual schedule. The cpu time for the ZIP program was 67 sec. (on an Amdahl 5860).

The Melbourne agreement specified an 8 hour minimum payment for a duty, with a maximum cost of 8:15. In fact, in the schedule found,

only one duty had a cost greater than 8 hours, so that this was very close to the optimal schedule, if not the optimum.

9.4.14 Summary

The preceding sections have described a variety of schedules which were compiled by the author using the IMPACS system at various stages of its development. At first, each new problem required new facilities to be introduced into the system, but it is now possible to tackle virtually any bus crew scheduling agreement without further modification, apart from writing the appropriate routines such as VALID and WAGES. As originally intended, the system has proved to be much more portable than the TRACS heuristic system. The schedulng exercises described above show that IMPACS is a general-purpose bus crew scheduling system which can produce good schedules for a wide range of problems, provided that they are not too large. Methods for dealing with large problems are discussed below in section 9.6. The next section describes the implementation of IMPACS for London Transport, when for the first time the system was handed over to schedulers for their own use.

9.5 Implementation For London Transport

London Buses Ltd., formerly London Transport, is by far the largest bus operator in the U.K., having between five and six thousand buses, on about 250 routes, and operated from about 60 garages. As well as being the largest operator in the country, its operations represent a considerable proportion of the total; for instance in 1980, of the 640 million bus miles run by local authority bus operators (comprising in addition to London Transport, the metropolitan Passenger Transport Executives, the District Councils and Scottish Regional Councils), London Transport was responsible for 27%.

However, the scheduling problems arising in London Transport are not particularly large: each route is scheduled separately, for both vehicles and crews, and many routes are operated by more than one garage, so that each crew schedule is relatively small. The largest crew schedule operating in 1984 had about 75 duties. Further, there is only one scheduling agreement for the whole of London Transport, and all the scheduling work is done by a central Schedules Office, which simplifies the introduction of a new system.

L.T. have developed a computerised bus scheduling system, called CABS, and in the 1970's some work was done on the development of a crew scheduling system, but little progress was made. The University of Leeds Operational Research Unit carried out a pilot study for London Transport

in 1974 using the heuristic crew scheduling system TRACS [59]. The results of the pilot study were not very satisfactory: the schedules produced for the three trial schedules were not as good as the manual schedules. The conclusion of the pilot study was that the program in its existing form could not produce good schedules for London Transport. It was suggested that a mathematical programming approach might be more suitable and in fact Manington [31] applied his mathematical programming methods to one of the pilot study problems, which required 20 duties. A few hundred possible duties were generated by hand, and the schedule produced was identical to the manual schedule. Further work, aimed both at modifying TRACS to give better results for London Transport and at developing the mathematical programming method, was proposed, but London Transport did not proceed.

Towards the end of the SERC project under which a prototype version of IMPACS was developed, the program was tried out on some test schedules from London Transport. The results seemed promising and were sent to the Schedules Manager for his comments. He felt that the program should be investigated further, and while the SERC project was running, more test schedules were compiled. Finally in August 1983, London Transport agreed to sponsor further development of the system.

IMPACS at this stage required further development work for three reasons: first, it had not been used by schedulers before, and so had to be made easier to use; secondly, it had to be tailored to the scheduling conditions of London Transport; and thirdly, the algorithms used needed to be improved, if possible, so that the running time of the program could be reduced.

It was agreed that a pilot study lasting up to eighteen months should be carried out, during which time IMPACS would be modified as just described. At the end of the development programme, it was intended that IMPACS should be capable of compiling virtually all L.T. schedules and of producing results which would be acceptable, as well as following the terms of the agreement. During the pilot study, it was planned that only one scheduler should use IMPACS, but if it was decided to adopt IMPACS permanently, he would then train the other schedules staff.

The Schedules Manager, in July 1984 [60], described the aims and benefits of IMPACS, as seen by London Transport at that time, as follows:

" (i) speeding up the production of duty schedules.

(ii) permitting the prompt and accurate evaluation of optional changes to the Union Agreement.

(iii) permitting the faster implementation fleetwide of a change to the Union Agreement.

(iv) allowing standardisation on an acceptable level of schedule efficiency.

(v) meeting union complaints more effectively with a prompt response and a variety of options where appropriate.

(vi) reducing the volume of data entry arising from the availability of the compiled schedule in computer media.

(vii) offering Management a basis for considering a more economic deployment of resources.

It can thus be seen that the production of more efficient duty schedules is not a specific aim, rather that existing standards are maintained, and duty schedules more speedily produced."

9.5.1 Modifications To IMPACS Input And Output

At the start of the IMPACS pilot scheme for L.T., the scheduler assigned to testing the system set up the data files in the standard form as described in the Appendix. Some modifications were introduced to make the creation of these files easier, for instance the addition of lines of text as a reminder of the significance of the data items. However, it soon became apparent that in order to make the system as easy as possible for L.T. schedulers to use, and to allow potential duties to be validated adequately, the standard forms of the files would have to be abandoned to some extent.

The main changes were required to the duty generation parameter file, and eventually a completely different format emerged for L.T., with a separate reading routine. This was because many of the items required in the standard form of the file are irrelevant to L.T.: for instance, there is no minimum paid day, and no maximum platform time, other than that implied by the maximum time on duty. The scheduler found it tedious to have to specify these items for every schedule. They are now omitted from the L.T. version of the file, and the reading routine supplies default values.

Changes were also made to parts of the bus schedule data file. The main part of the file, i.e. the list of relief opportunities on each bus, is common to both the L.T. version and the standard form, although for L.T. the direction in which the bus is travelling along the route can be specified for each relief opportunity; this then appears on the schedule listing, and also allows the scheduler to specify that at certain times of day reliefs in one direction or the other should be avoided if possible. The relief information, including directions of travel, is set up automatically by L.T.'s CABS bus scheduling system.

The arrays at the end of the bus schedule data file, listing minimum mealbreaks, etc., as described in the Appendix, could not be used for

those L.T. schedules in which the travel time between relief points varies through the day. This generally happens when there is a relief point at some distance from the garage, and the crew are required to catch a bus on another route from the relief point to the garage or vice versa, in order to sign on or off or to take a mealbreak in the garage canteen. Provided that the bus service is sufficiently frequent, a travel time allowance is given, based on the frequency and the running time, plus any walking time before or after the bus journey. The travel time will vary through the day, as the frequency and running time of the buses change.

Variable travel times affect all the arrays at the end of the bus schedule file, since mealbreaks and joinups starting and finishing at different points involve travel time, and so do signing on and off allowances at points away from the garage. In L.T. schedules, the mealbreak is unpaid apart from the travel time, so the paid mealbreak array is also affected.

A further complication is that although there is always a canteen at the garage, many L.T. crew schedules also make use of other canteens, which may have different closing times from the garage canteen; sometimes, mealbreaks at a relief point away from the garage may be taken at a local canteen for part of the day, and then at the garage after the local canteen closes, so that even if the travel time is constant, the minimum mealbreak may vary through the day.

Initially, these problems were avoided by restricting the use of IMPACS to schedules in which the travel time was constant and each relief point used only one canteen, but early in 1984 an L.T. version of the bus schedule data file was introduced. The arrays at the end of the file were replaced by travel time information, including variations if any, and, for each relief point, details of the nearest available canteen (varying by time of day if necessary) and the minimum time to be allowed at the canteen.

If there is no variation through the day, the reading routine constructs the minimum mealbreak array and so on from this information. Otherwise, whenever an item such as minimum mealbreak is required, its actual value at the particular time of day must be calculated.

As mentioned in chapter 8, the list of duties is produced in a specially-designed layout for L.T., to match as closely as possible an existing document. The graphical form of schedule output has also been redesigned to L.T.'s specifications, to correspond to the "cutting-up sheet", which is the working document used by the schedulers. Finally, a

duty summary is produced, which gathers together a few important measures of the quality of the schedule, mainly for comparison with the previous schedule.

9.5.2 Nominated Travel

If the bus service between the relief point and the garage is not sufficiently frequent for a travel time allowance to be acceptable, then a particular bus on which the crew can travel must be nominated. This is a more difficult problem to deal with, but has been incorporated into IMPACS for L.T.

An extra data file must be set up listing all possible bus trips which can be used for travel from the relief point to the garage and vice versa, together with the walking time required at each end. Whenever the travel time between the two points is required, for instance to calculate the signing on allowance at the relief point at a particular time of day, the list is searched to find the best bus trip.

An extra output file is also produced, listing the nominated trips and the duties in which they are required.

Setting up the list of possible trips is tedious for the scheduler, but fortunately nominated travel affects only a small proportion of schedules, and as it only occurs when the frequency of buses between the relief point and the garage is four buses per hour or less, the number of possible trips is limited.

9.5.3 Penalty Costs

The penalty costs attached to certain kinds of duty are calculated by a company-specific routine, as described in section 9.2.3. For London Transport, this routine reads its own penalty cost parameter file, so that the weights assigned to various undesirable features can be varied by the scheduler. This is the only case in which penalty costs have been discussed in detail with schedulers, but it is likely that a similar arrangement would often be advantageous. By varying the values of the parameters, the scheduler can if necessary produce more than one version of the same schedule, and for instance allow for variations in preferences between different garages.

The first part of the file is also read by the L.T. VALID routine, to cater for a clause in the union agreement that the work content of early-starting and late-finishing duties should not be increased from schedule to schedule. The exact meaning of this clause had not previously been expressed in a form suitable for incorporation into IMPACS, but it was eventually agreed that the scheduler should specify the early-starting

and late-finishing duties concerned, and the desired spreadover for each duty (usually the spreadover of the corresponding duty in the current schedule). The scheduler can then specify the maximum amount by which the spreadover can exceed the desired spreadover, which is used by the VALID routine, and the penalties for deviating from the desired spreadover in either direction.

An example of a penalty cost parameter file is shown in Figure 9.1. The penalties represent extra costs in minutes to be added to the other costs of a duty, and they are cumulative, so that a duty with several undesirable features (such as a three-bus spreadover duty) could incur a large total penalty. (A split duty is called a spreadover duty in London Transport.) Some of the items reflect clauses of the union agreement, e.g. the number of three-bus duties should be kept to a minimum. Others relate to types of duty which are unpopular with drivers, although not mentioned in the agreement: for instance, the proportion of spreadover duties (spreadover > 8:40) is limited by the agreement and should be minimised, but short spreadover duties (spreadover 8:41 - 10:30) are especially unpopular and so carry a higher penalty.

Although there are many items in this data file, only a few of the penalties will be given large values at one time. Schedule compilers are aware from their own experience of compiling schedules manually that usually the only way of eliminating one type of undesirable feature without incurring more duties is to introduce another type of undesirable feature.

9.5.4 Changes To Running ZIP

To compile acceptable schedules for L.T. it is almost invariably necessary to include penalties for undesirable features in the costs of the duties. However, experience showed that occasionally this meant that minimising the total cost of the duties selected gave a solution with more than the minimum number of duties. A strategy for running the ZIP program was devised to ensure that a schedule with the minimum number of duties would be found and that if necessary duties with undesirable features would be used to achieve this.

Initially, the strategy required the scheduler to run the ZIP program more than once, first without penalty costs and solving only the relaxed L.P. to establish the minimum number of duties required, then constraining the number of duties to this minimum and finally adding the penalty costs and reoptimising, this time going on to find an integer solution. A new mode of running ZIP was later introduced which

```
NUMBER OF EARLY DUTIES TO BE LIMITED
2
BUS NUMBER OF EACH
8 10
CURRENT SPREADOVER OF EACH
759 747
NUMBER OF LATE DUTIES TO BE LIMITED
0
BUS NUMBER OF EACH

CURRENT SPREADOVER OF EACH

MAX. EXCESS ALLOWED OVER CURRENT SPREADOVER
30
PENALTY PER MINUTE OVER CURRENT SPREADOVER
10
PENALTY PER MINUTE UNDER CURRENT SPREADOVER
5
PENALTY FOR S SPELL (TIME IN CHARGE > 4:30)
100
EXTRA PENALTY  FOR S SPELL AT END OF DUTY
50
PENALTY FOR X SPELL (TIME ON BUS > 4:30)
50
PENALTY FOR 1ST SPELL ON BUS OVER MIDDAY (1200-1500)
100
PENALTY FOR 3-BUS DUTY
200
MIN. DESIRABLE SPELL LENGTH (MINUTES)
60
PENALTY FOR SHORTER SPELL
100
EXTRA PENALTY IF SHORT SPELL AT END OF 3-BUS DUTY
50
PENALTY FOR SPREADOVER GREATER THAN 10:30
300
PENALTY FOR SHORT SPREADOVER DUTY (SPREADOVER 8:41-10:30)
700
TIGHT RELIEFS TO BE PENALISED AT SPECIFIED TIMES
NONE
CHANGEOVERS IN SPECIFIED DIRECTIONS TO BE PENALISED
NONE
```

Figure 9.1 Example Of A London Transport Penalty Cost File

combined the stages of the multiple run into one: this is described in section 6.12.

The scheduler also pointed out that usually the number of duties required in the schedule is known in advance, and a schedule with more duties would be unacceptable. This is because the usual reason for introducing a new schedule, at least in present circumstances, is to reduce the cost of operating a route, either by reducing the service or by changing from two-person to one-person operation. The cost saving required, and hence the number of drivers or crews allocated to a route, is determined in advance. The number of duties is therefore a target figure rather than an estimate, and it may not be achievable with the current bus schedule. If necessary, work will be removed from the bus schedule until a crew schedule with the target number of duties can be compiled. Hence, if the sum of the duty variables in the solution to the relaxed L.P. is already too high, it is a waste of time to find an integer solution. An alternative strategy was introduced in which the scheduler can specify the target number of duties, and if this can be shown to be unattainable when the relaxed L.P. has been solved, the program stops. A later modification of this strategy used the fact that the minimum number of duties is already known (i.e. the specified target) to allow the penalty costs to be included from the start, as described in section 6.12.

9.5.5 London Transport's Experience

It was decided that during the development programme, from August 1983 to the end of 1984, only one scheduler should be assigned to testing IMPACS. However, from an early stage, the schedules used to test IMPACS were selected from the current Schedules Office work programme, and so were intended, if successfully compiled, to be put into operation. By October 1984, the scheduler was doing a great deal of operational work using IMPACS,for all six sections of the Schedules Office. Within one month, he compiled more than 35 different schedules, including one requiring 86 duties, which took just over 400 seconds (on L.T.'s own computer, an IBM 3033). IMPACS was already proving its usefulness, even with only one scheduler using the system. The scheduling sections were referring as much work to him as he could cope with; many of these jobs were either very urgent (for instance dealing with Union complaints about schedules which were due to be operational shortly), or were expected to be, or had already proved to be, difficult to compile.

As Elms [60] reported in July, 1984, "Although still in the development stage, we are already getting much practical use from the

system on experimental work in connection with possible changes to the union agreement, as well as with jobs where difficulties have been encountered during manual compilation."

Towards the end of the development programme, the training of the other schedulers began. The training schedule was subsequently held up by delays in the development of the bus scheduling system CABS, but by June 1985, all the schedulers had been trained to use both CABS and IMPACS. Since the beginning of 1985, they have been using IMPACS without outside assistance, and are very satisfied with the quality of the schedules produced.

When the development programme was instigated, it was envisaged by the then Schedules Manager that the IMPACS schedules would only be a first attempt, capable of considerable improvement and "polishing" by the scheduler. Even so, he felt that the system could be of benefit because schedules would be produced more quickly than before. However, experience during the development programme raised the schedulers' expectations of the quality of the IMPACS schedules, almost to the extent that they are surprised if the IMPACS schedule can be improved at all. For instance, one scheduler found that by making a five-way exchange of the duties in an IMPACS schedule, he could reduce the cost by 20 minutes, and wondered why IMPACS had not found the cheaper solution: investigation in fact showed that if the COMPARE program (described in section 5.10) had not been run, the cheaper solution would have been found, although the run would have taken considerably longer.

At the time of writing approximately 85% of London Buses' schedules are compiled using IMPACS. Of the remaining 15%, some are manually compiled by schedulers who have an antipathy to computer systems, and do not use either CABS or IMPACS: the rest are schedules which for various reasons are not covered by the standard scheduling agreement.

As an illustration of the usage of IMPACS by the Schedules Office, during the six months from July to December 1985, one of the six scheduling sections compiled 115 schedules using IMPACS. On average, each schedule required 2.2 submissions, to produce alternative solutions, and each submission took 195 seconds of cpu time. The Schedules Office is charged for computer time, since the computing department is now treated as a separate business, although the rates are probably lower than normal commercial charges; the computer time for each schedule cost on average £15. The elapsed time for each submission averaged 25 minutes: because IMPACS can produce schedules so much more quickly

than a manual scheduler, the cost of running the system is thought to be worthwhile. In fact, at busy times, the schedulers run IMPACS in the more expensive foreground system, rather than as a batch job, in order to get results back as soon as possible. Overall, the Schedules Manager feels that the cost of running IMPACS is outweighed by the benefit of obtaining good schedules quickly.

9.6 Solving Larger Problems

Running the ZIP program is the most time-consuming part of the IMPACS system. As the size of the set covering problem increases, as measured by the number of constraints and the number of variables, the time taken to find a solution to the relaxed L.P. increases, as does the time taken to solve each subproblem in the branch-and-bound tree. However, the total time taken to find a good integer solution also depends on the number of nodes of the branch-and-bound tree which have to be solved, and this does not depend in any obvious way on the size of the problem.

The methods described so far have proved to be successful for scheduling problems which yield set covering problems with a maximum of about 300 constraints and about 6000 variables, although for some problems within this range, it may be difficult to find an integer solution: the program may then spend a long time exploring many nodes of the branch-and-bound tree, without in some cases producing a schedule in the end. For instance, the Yorkshire Traction Rawmarsh schedule described in section 9.4.10 shows that if the proportion of three-bus duties is much higher than usual, the problem is likely to be difficult to solve. Conversely, if a high proportion of one-bus duties is allowed, as in many North American and European schedules, larger problems can be solved without undue difficulty, as is demonstrated by the Los Angeles schedule described in section 9.4.12. In general, however, within the limits of 300 constraints and 6000 variables IMPACS can compile schedules successfully.

Because the eventual number of constraints and variables in the set covering problem is unknown at the outset, the limits on the reasonable operating size of IMPACS are described in terms of an approximate upper limit of 80 on the number of duties in the schedule, with the proviso that if the relief times are very frequent, giving a large number of constraints and/or variables, the size of schedule which IMPACS can handle is correspondingly smaller.

For many bus companies an 80-duty limit is entirely adequate, or

covers a large proportion of the schedules, so that even if no schedule larger than 80 duties could be compiled, IMPACS could still be of considerable use. Very much larger problems than 80 duties have been compiled using IMPACS, however, by decomposing the original problem into two or more subproblems which are small enough to be solved individually.

The decomposition is done by allocating each running board to one of the subproblems. This is in contrast to the decomposition approach investigated by Ward, Durant and Hallman [25], described in section 2.3.4, in which the original problem is divided into smaller problems by time of day.

The decomposition has to be done carefully because otherwise the solution is likely to require more duties than would be necessary if the whole problem could be solved at once. A possible way of checking that the decomposition has not caused significant suboptimality would be to estimate the number of crews required to cover the whole bus schedule and each of the subproblems, but unfortunately for most agreements it is impossible to estimate the crew requirements with sufficient accuracy.

Five schedules have so far been compiled using this method; the procedure is still under development and has not yet been fully automated, but the results have been encouraging. In the four cases in which the number of duties in the current schedule is known, IMPACS produced a schedule with fewer duties. The decomposition technique has largely been developed by A. Wren and others.

9.6.1 Lothian Region Transport

This was the first schedule compiled using IMPACS which was too large to be solved directly and had to be decomposed into subproblems. It was initially commissioned by Lothian Region Transport (LRT) as a check on the efficiency of their existing schedules; this first exercise was carried out in January 1984. After some correction to the scheduling rules, the exercise was repeated in January 1985 with a view to implementing the IMPACS schedule. Both exercises were carried out by A. Wren.

The schedule covered all the work operated by one of LRT's garages; the manual schedule had 225 duties and 41 overtime pieces. In view of the approximate upper limit of 80 duties on the size of problems which can be solved directly using IMPACS, the bus schedule was divided into three approximately equal parts.

The division was done by examining the bus schedule to identify

buses which for some reason should be in the same subproblem. For instance, each bus worked a different route and the relief point on one route was remote from the home garage, in fact at another garage. Crews starting and finishing a mealbreak at this point could use the local canteen; otherwise, a long travel time was involved. Hence it was sensible to keep all the buses operating this route together.

The analysis also identified routes or groups of routes for which the frequency of relief times and the travel time between the relief point and the garage were such that efficient mealbreaks could be formed by transferring the crew from one bus to another within the group. These buses were also kept together within the same subproblem.

The most notable feature of the union agreement was that the duties were all very short: the maximum spreadover was 7:50 (which was one reason why mealbreaks should be as short as possible).

When the exercise was repeated in 1985, with some changes to the scheduling rules in IMPACS, a schedule with a total of 223 duties and 30 overtime pieces was produced, saving two full duties, eleven overtime pieces and nearly 65 hours paid time per day over the manual schedule. The exercise therefore demonstrated that decomposing large problems into smaller subproblems could be a successful way of extending the use of IMPACS.

A second, larger, schedule was compiled for LRT late in 1985, yielding 263 duties and 45 overtime pieces, compared to 263 duties and 65 overtime pieces in the manual schedule.

9.6.2 CTCUQ

In May 1984, the Quebec Urban Community Transit Commission (CTCUQ), which operates in Quebec City, sent details of their requirements for computer scheduling systems with an invitation to compile schedules for a test set of data. The crew scheduling requirements were for an interactive system for daily crew scheduling, and a more powerful system capable of optimising the quarterly crew schedules. The whole of the CTCUQ system is rescheduled every quarter, which involves compiling a crew schedule for more than 400 drivers on weekdays.

A fictitious set of data for a weekday quarterly schedule was sent, with a description of the crew scheduling rules, although as in other exercises, without being able to see an existing CTCUQ schedule, it was impossible to be certain that the rules had been correctly interpreted.

The schedule had about 300 peak buses and was expected to contain more than 400 duties, with up to 30 unassigned pieces of work. In view of this, it clearly had to be split up into smaller units for compilation by IMPACS. Six subproblems were formed by dividing the bus schedule as evenly as possible, so that the bus hours and also the profile of buses on the road at different times of day were approximately the same in all the subproblems. Because many of the duties were one-bus duties or trippers, it seemed that there was less likelihood of critical links between buses than in other schedules; the schedule also had to be produced very quickly, so that there was not enough time for a detailed analysis of the bus schedule.

The CTCUQ scheduling rules were unusual in specifying minimum lengths for a spell of work and the working time in a duty. The latter was equivalent to a minimum cost, since the mealbreaks were unpaid. Although the duty generation parameters include minimum cost and minimum spell length, these are to enable the scheduler to avoid forming unnecessary duties; they are rarely specified in the scheduling agreement. The minimum length of spell was 2:15, and the minimum working time 7:15. Duties could work on one, two or three buses, with a working time between 7:15 and 8:30, and the maximum spreadover was 10 hours.

Since the maximum length of a spell in a two- or three-bus duty was 6 hours, and the minimum length of a one-bus duty was 7:15, no spell of work could be between 6:00 and 7:15 long; also, spells less than 2:15 were not allowed, as already mentioned. These rules allowed many relief times to be eliminated. As in the Los Angeles exercise, because of the one-bus duties, the relief time selection program could not be used.

The one-bus duties were formed by a specially written routine, but since it appeared that the unassigned pieces should be on short peak-only buses, they were allowed to appear in the schedule as slack variables.

The six subproblems were solved separately, giving a total of 409 duties and 20 unassigned pieces of work. For this exercise, the heuristic improvement routines (SWAP and MOVE) which had so far been developed for L.T. schedules only, were adapted for the general case, and applied both to the individual schedules and to the combined schedule. The heuristics resulted in considerable cost savings, although the number of duties was not affected.

In the light of later experience, it would have been better to have carried forward some work from each subproblem to the next, and this might have reduced the number of duties or unassigned pieces. An

obvious choice for carrying forward would have been the unassigned pieces of work, since the number of these was to be minimised. Also in some of the subproblems, there was some over-cover, which was more difficult to get rid of than usual because of the minimum spell and minimum cost rules; this problem might have been overcome by carrying forward the affected duties.

In all the subproblems integer solutions were found relatively quickly, and in a much shorter time than required to solve the relaxed L.P. The total cpu time required for the whole exercise was about 26 minutes, on an Amdahl 470/V7. For a quarterly task, computer times of this order, or even longer times on a much slower machine, would probably be acceptable.

9.6.3 GMT Queens Road Depot

As a follow-up to the Bolton exercise, described above, Greater Manchester Transport asked that a further exercise should be done, involving a larger schedule. Since many GMT schedules are too large to be solved directly using IMPACS, they wanted to be sure that satisfactory results could be obtained by decomposing the problem. The schedule was the Saturday schedule for Queens Road depot, and the exercise was carried out by A. Wren in the summer of 1984; the current manual schedule had 197 duties. The process followed is described in a report to GMT [61].

The problem was divided into three subproblems, the first containing approximately 80 duties. It was planned to carry forward work from the solution to the first subproblem to add to the the second subproblem, and from the first two subproblems to add to the third. To allow for this, the three subproblems initially contained 39%, 33% and 28% respectively of the work to be covered.

Wren devised a procedure for allocating buses to subproblems which is suitable for running automatically, but because of the time constraints on the exercise, it was not programmed and a simpler procedure was actually carried out by hand.

Wren's procedure forms provisional early and late duties, the intention being that any duties which may be critical to the formation of a good schedule should appear amongst the set of provisional duties.

On every bus, the latest relief time at which the first crew can be relieved is marked. The first and last times on each bus are also marked. The marked times are put into one or both of two sets; a set of times at which spells of duty can start, and a set of potential finishing times. The

sets are sorted into chronological order.

Each starting time is considered in turn. If there are any finishing times which can form a link with this starting time, i.e. a crew can finish their first spell of duty at the finishing time, take a mealbreak and then start a second spell of duty at the starting time, the starting time and all the possible finishing times are starred, and a quality measure calculated for each potential duty: Wren suggested using the spreadover of the duty.

Since the first times in the set of starting times will be the starting times of early buses, they will be earlier than any of the times in the list of finishing times and so no links can be formed.

If a starting time is later than some of the finishing times but no link can be formed, links which only require a joinup are considered instead. The latest finishing time which can form a reasonable two-bus stretch using this link is chosen; the buses concerned are grouped together and the times are removed from both lists. A new marked time is found on the second bus at the end of the stretch and placed in both lists as a potential starting and finishing time.

Otherwise, a "best" set of links is formed between the potential finishing times and starting times, by solving a matching problem for small batches of potentially linked starting and finishing times, maximising the quality measure calculated for each duty.

This process stops when some cut-off time such as 1200 is reached in the set of starting times; the provisional assignment of early duties has then been formed.

The process is repeated in reverse to form the late duties; however, because in many schedules (though not the Queens Road schedule) it is important to ensure that as many as possible of the early and late duties meet, the construction of late duties must take account of the early duties already formed.

Finally, the buses are assigned to subproblems. Each link formed in the preceding analysis involves two buses, and most buses will be involved in more than one link. By assigning some kind of priority level to the links, buses can be grouped together until groups of roughly the right size to form the subproblems have been found.

As already mentioned, this process was not followed in full for the Queens Road schedule. None of the early duties could finish late enough to meet the starts of late duties, so that it was not necessary to look for links which would be critical for that reason. The only part of the process which was judged to be essential for this schedule was the part which

ensures that joinups in early duties can be formed when required.

Having formed the three subproblems on this basis, Wren compiled a schedule for each of the three independently. As it was planned to carry forward work to augment the second and third subproblems, it was unnecessary to solve them separately, but this was done partly to see how useful transferring work would be, if at all.

The three separate schedules had 75, 63 and 58 duties respectively, giving a total of 196, which was already a saving of one duty compared with the manual schedule.

A total of 20 duties were transferred from the first subproblem to the second. These included the five split duties, since the number of split duties was to be minimised. The most inefficient duties were also transferred, i.e. the duties with a low work content. The IMPACS schedule for the augmented second subproblem had 82 duties, saving a further duty.

Finally, 40 duties were transferred to the third subproblem, including the split duties from the augmented second subproblem and the duties with lowest work content from the remainder of the first subproblem and the augmented second subproblem.

The augmented third subproblem required 96 duties, giving a total of 193. The augmented problems can be rather larger than 80 duties, because the work transferred consists mainly of separate spells of duty which will not be split up by IMPACS, so that each gives rise to only one constraint in the set covering problem and the size of the problem does not increase in proportion to the amount of work transferred.

The table below shows the number of duties in each subproblem at each stage:

Subproblem	1	2	3	Total duties
Separate schedules	75	63	58	196
Move 20 duties from 1 to 2	55	82	58	195
Move 40 duties from 1 & 2 to 3	44	53	96	193

The exercise demonstrated that even if the allocation of running boards to subproblems is done with care, carrying forward duties from

the solution to one subproblem to amalgamate with a later subproblem can reduce the cost of the final schedule significantly, to the extent of reducing the number of duties required. This technique has therefore been adopted as part of the general method for solving large problems, as described below in section 9.6.5.

Compiling the three schedules which were actually necessary (subproblem 1, augmented subproblem 2, augmented subproblem 3) took 531, 465 and 574 seconds of cpu time respectively. The exercise was carried out on an Amdahl 470/V7.

To produce the final schedule, the separate schedules for the three subproblems were combined and the heuristic improvement techniques described in chapter 7 were used to reduce the total cost. The result was accepted by GMT as satisfying their conditions, and as already stated, it required four fewer duties than the manual schedule.

9.6.4 Tyne and Wear South Shields Depot

A test schedule was produced in November 1984 for Tyne and Wear P.T.E., covering the operations of their South Shields depot. The current schedule for this depot had 142 duties.

Although this was a Monday to Friday schedule, the maximum spreadover was too short to allow duties to cover both peaks. The split duties had a maximum spreadover of 9:15, compared with 8 hours for other types. The maximum driving time for any duty was 6:30, so that all the duties were relatively short.

The schedule was split into two approximately equal parts, but with some attention paid to the possibilities of linking split duties covering the latest morning peak buses with the earliest finishing late duties, so that no extra middle duties would be required to cover the evening peak.

The first half of the schedule had 563 pieces of work, which were reduced to 288 by the relief time selection process, and a crew schedule was compiled for this half with 72 duties.

The duties involving over-cover and the split duties were carried over to the second half of the schedule, leaving 58 duties. Only 9 split duties could be used in total, so that because the remaining 58 duties of the first schedule now had no split duties, this limit could be applied to the second half.

The second half of the schedule then had 667 pieces of work, of which 328 were selected, and gave a crew schedule with 82 duties, so that the combined schedule had 140 duties, a reduction of two duties

compared with the manual schedule. The total cpu time required to run the ZIP program was approximately 1000 seconds (on an Amdahl 470/V7).

9.6.5 General Method For Large Schedules

Although the method of using IMPACS to compile large schedules is still under development, a general method can be described in outline.

The first stage is to determine the number and size of the subproblems and the number of duties to be carried forward from the solutions to the earlier subproblems to be added to the later subproblems, in the fashion described in section 9.6.3.

The total solution time can be expected to increase as the size of the individual subproblems increases, because the time taken to compile a schedule using IMPACS increases faster than the size of the problem. On the other hand, if only a small proportion of the total work to be covered is considered in each subproblem, the overall quality of the schedule produced is likely to suffer. Since the difficulty of compiling schedules manually also tends to increase with the size of the schedule, with large schedules there is often more potential for making savings than with smaller schedules, and hence, within reasonable limits, it is worthwhile using a strategy aimed at producing a good quality IMPACS schedule, at the expense of computer time.

If each subproblem were to be solved independently, without carrying forward work from earlier subproblems to the later ones, the best quality schedule would be obtained by forming the smallest number of subproblems, all of approximately equal size. If 80 duties is taken as the operating limit of IMPACS, then if the whole schedule requires n duties (where n can be roughly estimated from the expected average work content of a duty), this would mean that the original bus schedule should be divided evenly into [n/80]+1 subproblems.

The idea of carrying forward work from one subproblem to another tends to conflict with the strategy just described. It is clear that if a final subproblem were set up, consisting of duties selected from each of the schedules for the existing subproblems, the quality of the solution is likely to improve. In theory, if computer time were no object, this could be repeated indefinitely, selecting a different subset of duties each time, and compiling a new schedule to cover the same work; the new schedule would be likely to be a better way of covering the work, for at least a few more iterations. If these ideas are applied when dividing the original problem into subproblems, the later subproblems should be small, to allow a large number of duties to be carried forward from the earlier

subproblems; the first subproblem, however, should be as large as possible.

A possible compromise between these two strategies, the first of which divides the original problem into the smallest number of subproblems, and the second of which requires one large subproblem and many small subproblems, is to create one or two extra subproblems, beyond the minimum required (i.e. [n/80]+1) in order to allow a reasonable amount of work to be carried forward. The first subproblem should then be as large as possible (i.e. roughly 80 duties' worth of work) and the rest of the work should be divided between the remaining subproblems, in such a way that successive subproblems contain less and less of the original bus schedule, in order to allow more room for duties to be carried forward. In fact, as mentioned in section 9.6.3, since the transferred work often consists of individual spells of duty, it is easier for IMPACS to deal with than the original running boards, so that the later subproblems can be somewhat larger than 80 duties when augmented by the duties carried forward.

Having decided on the size of each subproblem the next stage is then to allocate the work to the subproblems. The general method outlined in section 9.6.3 could be followed, although this might need to be modified for particular cases.

The solution of the individual problems is straightforward, apart from the transfer of work to later subproblems. The duties which should be transferred include:

a) duties with over-cover;

b) duties of any type which are to be minimised;

c) duties with high penalty costs;

and finally,

d) the most inefficient duties, up to the total number required to be transferred.

An appropriate measure of efficiency may be the work content of the duty, or if the mealbreaks are paid, some combination of the work covered and the cost of the duty. Whereas the duties in the first two categories will only be carried forward from one subproblem to the next, the worst or most inefficient duties should be selected from all the previous subproblems considered together.

There are still some questions concerning carried forward work which will be resolved when more experience of solving larger problems

has been gained. For instance, the number of duties to be transferred from the earlier subproblems is decided at the start, before any subproblems are solved, so that although ideally all duties with over-cover and all duties of any type to be minimised should be carried forward, sometimes this may not be possible. In the case of over-cover, a possible way of dealing with this is to edit the duties involved manually and if possible transform pairs of overlapping duties into an efficient duty, to be left in the original subproblem, and a short duty, to be transferred. If there is an upper limit on the number of duties of a particular type, such as split duties, and it is impossible to carry forward all the duties of this type, the worst duties (by an efficiency or quality criterion) should be selected for transfer. However, because some duties of the restricted type will then be left behind in the successive subproblems, the number of such duties in the final subproblem may have to be severely limited, which may cause difficulties.

Transferring duties from one subproblem to another was at first a tedious process involving editing the data files manually. Recently, however, an interactive program has been written which allows the scheduler to describe the duties to be carried forward: this is now being used by Greater Manchester Transport. (The program was not written by the author.)

Having compiled each subproblem separately, a schedule for the total problem is obtained by combining the duties from the final subproblem with the duties not carried forward from the previous subproblems. The heuristic improvement routines described in chapter 8 can then be applied to the combined schedule, and may result in considerable cost savings, although the total number of duties will not be affected. It is important not to use these improvement techniques on the individual subproblems before duties are transferred, because evening out the quality of the duties is regarded as beneficial even if there is no cost saving, so that the most inefficient duties would tend to be changed before they can be identified and transferred to another subproblem.

CHAPTER 10. CONCLUSIONS

A crew scheduling system, IMPACS, has been developed, which has been demonstrated to be successful for a wide variety of scheduling conditions. The crew scheduling problem is formulated as an integer linear programme, using a formulation which is an extension of set covering.

The formulation requires that a very large set of possible duties should be generated, from which the duties forming the schedule are to be selected in such a way as to minimise the total cost. In principle, all possible legal duties should be generated, but for realistic problems this is impossible. A number of heuristic techniques and short-cuts are used in the system, which reduce the set covering problem to a manageable size or reduce the solution time, but at the same time try to ensure that a good quality schedule is obtained. These are summarised below:

a) the bus schedule is analysed and many of the relief opportunities are banned, in the sense that the candidate duties generated cannot use the banned relief opportunities to start or finish a spell of duty (chapter 4);

b) the duties generated are further restricted, partly by the system and partly by parameters specified by the scheduler, who can thereby ensure that duties which are inefficient are not formed (chapter 5);

c) the set of candidate duties is considered as a whole and any duty which is completely included in another duty, but is no cheaper than the longer duty, is rejected; sometimes it may be appropriate to reject the shorter duty even if the longer duty is more expensive, but this increases the likelihood of over-cover (section 5.11.2);

d) the candidate duties are considered in quality order (worst first) and a duty is rejected if every piece of work which it covers is also covered by at least a specified number of other duties (section 5.11.3);

e) after the solution to the relaxed L.P. has been found, any duty which uses a relief time other than those used by the duties represented by the continuous L.P. solution is not considered further (section 7.3);

f) the program does not attempt to find an optimal integer solution, but terminates as soon as a good integer solution is found (section 7.5);

g) after a schedule has been found, a number of heuristic improvement techniques are applied (chapter 8).

By using these techniques, schedules requiring up to about 80 duties can be successfully compiled in an acceptable time.

Recent developments of the IMPACS system, which have mainly been carried out by A. Wren and others, have been aimed at devising methods for solving larger problems by decomposing the original bus schedule into smaller subproblems, as described in section 9.6.

In the future, different approaches to the decomposition of larger problems may be investigated, such as the division by time of day suggested by Ward, Durant and Hallman [25]; some work on this approach has already been done by Greater Manchester Transport, who are currently using IMPACS. The feasibility of running IMPACS on a micro such as an IBM PC-XT is also being considered.

The set covering formulation can be used to model many scheduling problems. However, the heuristic techniques incorporated into IMPACS which are described above have been designed specifically for bus crew scheduling problems, so that IMPACS is of limited use in other fields. The crew scheduling problems which arise in other public transport modes are in some respects very similar to bus crew scheduling problems, and one might imagine that IMPACS could be used for these problems. For instance, as discussed in chapter 2, air crew scheduling seems similar in many ways to bus crew scheduling, but the methods which have been devised for air crew scheduling problems would not be successful for bus crew scheduling, so that it seems clear that in spite of the superficial similarities, the problems are in practice sufficiently different to require different methods of solution. Rail crew scheduling problems again have many similarities to bus crew scheduling, and in fact a feasibility study of the application of IMPACS for the London Underground is currently being carried out. However, the Underground schedules present a number of difficulties, chiefly due to the fact that there are many possible crew depots, and it is not yet clear whether IMPACS will be appropriate for these problems.

Many fields other than public transport give rise to personnel scheduling problems in which an overall schedule must be devised, consisting of individual work-days subject to agreed conditions, to meet the required staffing levels, which may vary through the day. However, whereas public transport crews, because they move around with their vehicle, can only hand over to another crew at specific times, most other scheduling problems are based in a static environment and personnel can change over at any time. Hence, although the problems are similar in outline, they require very different methods of solution. Static personnel scheduling problems are in fact very like the HASTUS-macro model described in section 2.4.3 in which the possible relief times are

temporarily forgotten, and a schedule simply providing the required number of crews in each time period is found.

The HASTUS system is the only other crew scheduling system which has been adopted by any British bus company. In a number of recent cases, bus companies have considered both IMPACS and HASTUS before deciding which system to adopt; the choice has been made on the basis of test schedules compiled using both systems. The bus companies have not always disclosed the results, but it is understood that in most cases IMPACS produced the better schedule. However, in these trials it is not usually possible to see the existing manual schedules and discuss them with schedulers; in these circumstances it is difficult to be certain that the scheduling conditions have been properly understood, and hence the trials may not be a fair comparison of the two systems. In at least one case, because the bus company had not specified their requirements clearly enough, the HASTUS schedule was the better result according to one criterion (and the bus company decided in favour of HASTUS), but by the criterion which had been used in IMPACS, and had earlier been approved by the bus company, the IMPACS schedule was the better result.

As described in section 9.5, the IMPACS system has been in everyday use by the schedulers of London Buses Ltd. for more than a year; it is used to compile nearly all their crew schedules, producing good results much faster than was previously possible. Greater Manchester Transport have been using the system for some months on their own computer. The system is not yet in general use by schedulers, since many GMT schedules are very large and the decomposition of large problems for IMPACS is still to some extent under development. However, as in the case of the London Transport development programme, the person assigned to guide the introduction of IMPACS has from the beginning been using the system to compile real schedules, intended to be put into operation. Cleveland Transit have been using IMPACS since September 1985, so far mainly to investigate a wide range of possible changes to the union agreement; by using IMPACS, they have been able to negotiate a new agreement which they feel will be better suited to operating conditions following deregulation.

Although for users who have to pay for computer time, IMPACS can be expensive to run, it can produce good schedules much more quickly than a manual scheduler, and in some cases the schedules are better than a manual scheduler would produce. For the bus companies who are using the system, the costs are outweighed by these benefits.

During the development of IMPACS, schedules from several other

British and foreign bus companies were compiled, with results which were at least as good as the existing schedules, and some of these schedules were implemented. There seems no reason why IMPACS should not be able to produce good schedules for virtually all British bus companies, and since the scheduling conditions elsewhere are often simpler, in allowing a high proportion of one-bus duties, IMPACS can reasonably be claimed to be a general bus crew scheduling system.

REFERENCES

1. **Turner, J.F.** Timetable and duty schedule compilation (Road passenger transport). Sir Isaac Pitman & Sons Ltd., 1946.

2. **Hartley, T. and Wren, A.** Two complementary bus scheduling programs. In Computer Scheduling of Public Transport 2, ed. J.-M. Rousseau, pp. 345-369. North-Holland Publishing Company, 1985.

3. **Wren, A. (ed.)** Computer Scheduling of Public Transport, North-Holland Publishing Company, 1981.

4. **Rousseau, J.-M. (ed.)** Computer Scheduling of Public Transport 2, North-Holland Publishing Company, 1985.

5. **Young, A. and Wilkinson, J.C.** The scheduling of buses and crews by computer. Paper presented to the Public Transport Association annual conference, Scarborough, 1966.

6. **Elias, S.E.G.** The use of digital computers in the economic scheduling for both man and machine in public transportation. Special report no. 49, Kansas State University Bulletin, 1964.

7. **Elias, S.E.G.** A mathematical model for optimizing the assignment of man and machine in public transit "run-cutting". Research Bulletin no. 81, West Virginia University Engineering Experiment Station, 1966.

8. **Ward, R.E. and Deibel, L.E.** The advancement of computerized assignment of transit operators to vehicles through heuristic programming techniques. Presented at the Joint National Meeting of the Operations Research Society of America, the Institute of Management Sciences, and the Systems Engineering Group of the American Institute of Industrial Engineers, Atlantic City, 1972.

9. **Parker, M.E. and Smith, B.M.** Two approaches to computer crew scheduling. In Computer Scheduling of Public Transport, ed. A. Wren, pp. 193-221. North-Holland Publishing Company, 1981.

10. **Wilhelm, E.B.** Overview of the RUCUS package driver run cutting program - RUNS. Presented at the Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services, Chicago, 1975. (Unpublished).

11. **Goeddel, D.L.** An examination of the run-cutting and scheduling (RUCUS) system - a case analysis. Presented at the Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services, Chicago, 1975. (Unpublished).

12. **Landis, M.G.** A perspective on automated bus operator scheduling: five years' experience in Portland, Oregon. In Computer Scheduling of Public Transport, ed. A. Wren, pp. 61-67. North-Holland Publishing Company, 1981.

13. **Hildyard, P.M. and Wallis, H.V.** Advances in computer assisted runcutting in North America. In Computer Scheduling of Public Transport, ed. A. Wren, pp. 183-192. North-Holland Publishing Company, 1981.

14. **Luedtke, L.K.** RUCUS II: A review of system capabilities. In Computer Scheduling of Public Transport 2, ed. J.-M. Rousseau, pp. 61-116. North-Holland Publishing Company, 1985.

15. **Ball, M.O., Bodin, L.D. and Greenberg, J.** Enhancements to the RUCUS2 crew scheduling system. In Computer Scheduling of Public Transport 2, ed. J.-M. Rousseau, pp. 279-293. North-Holland Publishing Company, 1985.

16. **Howard, S.M. and Moser, P.I.** IMPACS - A hybrid interactive approach to computerized crew scheduling. In Computer Scheduling of Public Transport 2, ed. J.-M. Rousseau, pp. 211-221. North-Holland Publishing Company, 1985.

17. **Wren, A., Smith, B.M. and Miller, A.J.** Complementary approaches to crew scheduling. In Computer Scheduling of Public Transport 2, ed. J.-M. Rousseau, pp. 263-278. North-Holland Publishing Company, 1985.

18. **Martello, S. and Toth, P.** A heuristic approach to the bus driver scheduling problem. Eur. J. Opl Res. 24, 106-117, 1986.

19. **Carraresi, P., Gallo, G. and Rousseau, J.-M.** Relaxation approaches to large scale bus driver scheduling problems. Transpn Res., 16B, 383-397,1982.

20. **Heurgon, E.** Preparing duty rosters by computer. Presented at the Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services, Chicago, 1975. (Unpublished).

21. **Borret, J.M.J. and Roes, A.W.** Crew scheduling by computer: A test on the possibility of designing duties for a certain bus line. In Computer Scheduling of Public Transport, ed. A. Wren, pp. 237-253. North-Holland Publishing Company, 1981.

22. **Mitra, G. and Darby-Dowman, K.** A computer based crew scheduling system using a mathematical programming approach and experience of its use. Proceedings of the 13th Annual Seminar on Public Transport Operations Research, University of Leeds, 1981.

23. **Mitra, G. and Darby-Dowman, K.** CRU-SCHED - A computer based bus crew scheduling system using integer programming. In Computer Scheduling of Public Transport 2, ed. J.-M. Rousseau, pp. 223-232. North-Holland Publishing Company, 1985.

24. **Mitra, G. and Welsh, A.P.G.** A computer based crew scheduling system using a mathematical programming approach. In Computer Scheduling of Public Transport, ed. A. Wren, pp. 281-296. North-Holland Publishing Company, 1981.

25. Ward, R.E., Durant, P.E. and Hallman, A.B. A problem decomposition approach to scheduling the drivers and crews of mass transit systems. In Computer Scheduling of Public Transport, ed. A. Wren, pp. 297-312. North-Holland Publishing Company, 1981.

26. Ryan, D.M. and Foster, B.A. An integer programming approach to scheduling. In Computer Scheduling of Public Transport, ed. A. Wren, pp. 269-280. North-Holland Publishing Company, 1981.

27. Foster, B.A. and Ryan, D.M. An integer programming approach to the vehicle scheduling problem. Opl Res. Q., 27, 367-384, 1976.

28. Piccione, C., Cherici, A., Bielli, M. and La Bella, A. Practical aspects in automatic crew scheduling. In Computer Scheduling of Public Transport, ed. A. Wren, pp. 223-235. North-Holland Publishing Company, 1981.

29. Kregeloh, H. and Mojsilovic, M. Automated formation of staff schedules and duty rosters. Presented at the Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services, Chicago, 1975. (Unpublished).

30. Hoffstadt, J. Computerized vehicle and driver scheduling for the Hamburger Hochbahn Aktiengesellschaft. In Computer Scheduling of Public Transport, ed. A. Wren, pp. 35-52. North-Holland Publishing Company, 1981.

31. Manington, P.D. Mathematical and heuristic approaches to road transport scheduling. Ph.D. thesis, University of Leeds, 1977.

32. Lessard, R., Rousseau, J.-M. and Dupuis, D. Hastus I: A mathematical programming approach to the bus driver scheduling problem. In Computer Scheduling of Public Transport, ed. A. Wren, pp. 255-267. North-Holland Publishing Company, 1981.

33. Rousseau, J.-M. and Blais, J.-Y. HASTUS - An interactive system for buses and crew scheduling. In Computer Scheduling of Public Transport 2, ed. J.-M. Rousseau, pp. 45-60. North-Holland Publishing Company, 1985.

34. Rousseau, J.-M., Lessard, R. and Blais, J.-Y. Enhancements to the HASTUS crew scheduling algorithm. In Computer Scheduling of Public Transport 2, ed. J.-M. Rousseau, pp. 295-310. North-Holland Publishing Company, 1985.

35. Mitchell, R. Results and experience of calibrating HASTUS-MACRO for work rule cost estimation at the Southern California Rapid Transit District, Los Angeles. In Computer Scheduling of Public Transport 2, ed. J.-M. Rousseau, pp. 119-136. North-Holland Publishing Company, 1985.

36. Ball, M.O., Bodin, L.D. and Dial, R. Experimentation with a computerized system for scheduling mass transit vehicles and crews. In Computer Scheduling of Public Transport, ed. A. Wren, pp. 313-334. North-Holland Publishing Company, 1981.

37. Ball, M.O. and Bodin, L.D. A matching based heuristic for scheduling mass transit vehicles and crews. Transpn Sci. 17, 4-31, 1983.

38. Arabeyre, J.P., Fearnley, J., Steiger, F.C. and Teather, W. The airline crew scheduling problem: A survey. Transpn Sci., 3, 140-163, 1969.

39. Rubin, J. A technique for the solution of massive set covering problems, with applications to airline crew scheduling. Transpn Res., 7, 34-48, 1973.

40. Bodin, L., Golden, B., Assad, A. and Ball, M. Routing and scheduling of vehicles and crews: The state of the art. Comput. Opns Res., 10, no. 2 (Special issue), 1983.

41. Marsten, R.E. An algorithm for large set partitioning problems. Mgmt Sci. 20, 774-787, 1974.

42. Marsten, R.E. and Shepardson, F. Exact solution of crew scheduling problems using the set partitioning model: Recent successful applications. Networks, 11, 165-177, 1981.

43. Marsten, R.E., Muller, M.R. and Killion, C.L. Crew planning at Flying Tiger: A successful application of integer programming. Mgmt Sci., 25, 1175-1183, 1979.

44. Gerbracht, J. A new algorithm for very large crew pairing problems. Proceedings of the 18th AGIFORS Symposium, Vancouver, 1978.

45. Baker, E.K., Bodin, L.D., Finnegan, W.F. and Ponder, R.J. Efficient heuristic solutions to an airline crew scheduling problem. AIEE Trans., 11, 79-85, 1979.

46. Baker, E.K. and Fisher, M. Computational results for very large air crew scheduling problems. OMEGA, 9, 613-618, 1981.

47. Baker, E.K. Efficient heuristic algorithms for the weighted set covering problem. Comput. Opns Res., 8, 303-310, 1981.

48. Ellis, J.A. and Savin, H. Crew scheduling on railways, with particular reference to the local area problem. Presented at the 13th Annual Seminar on Public Transport Operations Research, University of Leeds, 1981.

49. Tykulsker, R.J., O'Neil, K.K., Ceder, A. and Sheffi, Y. A commuter rail crew assignment/work rules model. In Computer Scheduling of Public Transport 2, ed. J.-M. Rousseau, pp. 233-247. North-Holland Publishing Company, 1985.

50. Smith, B.M. A linear programming approach to crew scheduling. Report ULORU-29, University of Leeds Operational Research Unit, 1979.

51. Ryan, D.M. ZIP - a zero-one integer programming package for scheduling. Report CSS 85, AERE, Harwell, Oxfordshire, 1980.

52. Garfinkel, R.S. and Nemhauser, G.L. Integer Programming. John Wiley and Sons, 1972.

53. Reid, J.K. Fortran subroutines for handling sparse linear programming Bases. AERE report R-8269, AERE, Harwell, Oxfordshire, 1976.

54. Balas, E. and Ho, A. Set covering algorithms using cutting planes, heuristics and subgradient optimization: a computational study. Mathl Progm. Study, 12, 37-60, 1980.

55. Harris, P.M.J. Pivot selection methods of the Devex LP code. Mathl Progm. Study, 4, 30-57, 1975.

56. Goldfarb, D. and Reid, J.K. A practicable steepest-edge simplex algorithm. Mathl Progm., 12, 361-371, 1977.

57. Bellmore, M. and Ratliff, H.D. Set covering and involutory bases. Mgmt Sci., 18, 194-206, 1971.

58. Benichou, M., Gauthier, J.M., Hentges, G. and Ribiere, G. The efficient solution of large-scale integer programming problems. Mathl Progm., 13, 280-322, 1977.

59. Wren, A. Scheduling by computer - A report on a pilot study undertaken for London Transport. Report ULORU-15, University of Leeds Operational Research Unit, 1975.

60. Elms, J.M. Towards the fully computerised schedules office. Presented at the 16th Annual Seminar on Public Transport Operations Research, University of Leeds, 1984.

61. Wren, A. Report on Queens Road Saturday duties for GMT, 1984.

GLOSSARY

(Note: this describes the author's use of terms, and is not intended to be a general guide to crew scheduling terminology. Since the same term often means different things in different bus companies, even within the U.K., it would be difficult to compile a general glossary.)

BLOCK: The North American term for RUNNING BOARD.

DUTY: A day's work for a bus crew or driver.

JOINUP: A short break between two SPELLS of duty, allowing time to transfer from one bus to another.

LAYOVER: A period between the arrival of a bus at the terminus at the end of a journey and its departure on the next journey.

MEALBREAK: A rest period off the bus during a DUTY to allow the crew to have a meal.

n-BUS DUTY (n=ONE, TWO, etc.): A DUTY with n SPELLS.

ONE PERSON OPERATION (OPO): Operation of a bus by a driver only, without a conductor.

OVERTIME PIECE: A SPELL of work which is scheduled to be covered by a crew in addition to their normal DUTY, or on a rest day, and usually paid at a higher rate.

PAID THROUGH: A DUTY is PAID THROUGH if the whole of the SPREADOVER is paid time.

PLATFORM TIME: The time on bus in a DUTY, composed of driving time and LAYOVER time.

RELIEF POINT: A point where the crew may hand over the bus to another crew.

RELIEF TIME: A time during the day when the bus passes a RELIEF POINT.

RUN: The North American term for DUTY.

RUNNING BOARD: A sequence of trips assigned to one bus, starting with a departure from the garage and finishing with a return to the garage.

SIGNING ON ALLOWANCE: The time allowed for the crew to report for duty before starting work.

SIGNING ON TIME: The time at which the crew signs on: this is the starting time of the DUTY.

SIGNING OFF ALLOWANCE: The time allowed for a crew to sign off after leaving their last bus.

SIGNING OFF TIME: The time at which the crew signs off: this is the

finishing time of the DUTY.

SPELL: A period of time spent continuously on one bus during a DUTY.

SPLIT DUTY: A type of DUTY which has a longer SPREADOVER than other duties.

SPREADOVER: The length of time between the SIGNING ON TIME of a DUTY and the SIGNING OFF TIME.

SPREADOVER DUTY: A London Transport term, synonymous with SPLIT DUTY.

STRAIGHT DUTY: A DUTY which is not a SPLIT DUTY. (U.K.)

STRAIGHT RUN: A North American term for a ONE-BUS DUTY.

STRETCH: The period from the SIGNING ON TIME to the start of the MEALBREAK, or from the end of the MEALBREAK to the SIGNING OFF TIME.

TRIPPER: A North American term for an OVERTIME PIECE.

TWO PERSON OPERATION (TPO): Operation of a bus by a crew consisting of a driver and a conductor.

Note on times: times of day are written as four-digit numbers, using the 24-hour clock, so that 0700 is 7 a.m., 1900 is 7 p.m. Periods of time are written as hours and minutes, separated by a colon, so that 1:15 means one hour, fifteen minutes.

APPENDIX. IMPACS DATA FILES

(Adapted from the IMPACS manual)

A.1 Bus Schedule Data File

This file is the basic data file which is used by all the programs in the IMPACS system. It contains the information about when and where each bus can be relieved. It describes the relief points and the travelling time allowances, etc., required. An example of a bus schedule data file is given at the end of the section.

1. The first line has a heading of up to 50 characters which will be printed on the output.

2. The second line has the number of relief points.

3. 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 format of these lines is important, i.e. the correct spacing of the items.

The first character is a code for the relief point to be used in graphical output. The character can be a letter or a one-digit number or any other character. This is followed by one space, and then the name of the point to be used in the duty listing. The name may have up to twenty characters.

4. The following lines give for each running board, the bus running number and then for each relief opportunity, the time and relief point number. The bus running number always starts in column 1; continuation lines, listing further relief times and relief points on the same bus, are indented.

If a bus is out of service for part of the day, in the garage or at a bus park, this should be entered as two running boards (which may have the same running number).

The bus running number can only be a number, of up to 5 digits, and must not include letters.

Relief times should be hours and minutes, and shown as a number without any punctuation. Times after midnight should be shown as 2401, 2501, etc.

The relief point number corresponds to the order in the list of relief points given earlier, e.g. relief point 1 is the first in the list (the garage).

It is possible to distinguish between the arrival and departure at a relief point, when it is not necessary to allocate a crew to the period between the arrival and departure, by putting zero for the relief point following the departure time. This will be taken to mean that this is the departure, and the previous relief time is the arrival time.

Normally, relief times must be consecutive, but a departure from the garage can be before the corresponding arrival at the garage. This is to allow for the case where a change of crews also involves a change of bus, so that the relieving crew brings a new bus out of the garage and the crew being relieved drive their bus back. For instance, if the crews change over at 0900 and the driving time to the garage is 5 minutes each way, this would appear in the data as:

0905 1 0855 0

5. 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.

In the example, this is a block of three lines, each with three items, to correspond to the three relief points:

Starting at:	Finishing at:	GARAGE	ARCHWAY STATION	ARCHWAY STN STAND
	GARAGE	46	50	51
	ARCHWAY STATION	50	42	44
	ARCHWAY STN STAND	51	44	42

So that, for example, if a mealbreak starts at the garage and finishes at point 2 (Archway Station) or vice versa, the minimum mealbreak is 50 minutes.

6. The next block of data, the same size as in item 5, gives the portion of the mealbreak which is paid. (However, this is only used by the WAGES routine, which assigns a cost to every duty, and as this is specially written for each agreement, the array may sometimes be differently defined.)

7. Similarly, the next block gives the minimum joinup time required between each pair of points.

8. Finally, the signing on and off allowances at each relief point are listed (two lines with, for three relief points, three items on each line).

The lines describing the content of this section of the file, e.g. "MINIMUM MEALBREAK", are part of the file but are skipped over when the file is being read by IMPACS, except that the "MINIMUM MEALBREAK" line must begin with "M". They are put in as a reminder to the person setting up the file and can be abbreviated if desired.

Example Of A Bus Schedule Data File

```
ROUTE 104  HOLLOWAY GARAGE  MONDAY-FRIDAY
3
G GARAGE
A ARCHSTN
B A STN S
231 0451 1 0516 2 0623 2 0720 2 0821 2 0937 2 1027 2 1139 2 122
       1339 2 1427 2 1539 2 1644 2 1757 2 1856 2 1956 2 2039 3
       2226 2 2309 3 2400 1
232 0500 1 0556 2 0637 2 0735 2 0836 2 0959 2 1047 2 1159 2 124
       1359 2 1447 2 1558 2 1659 2 1812 2 1911 1
233 0649 1 0750 2 0851 2 1019 2 1107 2 1219 2 1307 2 1419 2 150
       1619 2 1714 2 1827 2 1918 1
234 0530 1 0626 2 0715 2 0820 2 0923 2 1039 2 1127 2 1239 2 132
       1439 2 1527 2 1639 2 1729 2 1852 2 1940 1
235  0643 1 0736 2 0850 2 0947 2 1059 2 1147 2 1259 2 1347 2 14
       1546 2 1657 2 1802 2 1916 2 1957 2 2056 2 2139 3 2227 2
       2410 1
236 0600 1 0656 2 0751 2 0905 2 1007 2 1119 2 1207 2 1319 2 140
       1519 2 1614 2 1727 2 1822 1
237 0613 1 0700 2 0805 2 0906 2 1025 1
237 1509 1 1559 2 1712 2 1822 2 1926 2 2009 3 2057 2 2156 2 22
       2327 2 2410 1
238 0706 1 0806 2 0920 2 1015 1
238 1526 1 1629 2 1742 2 1838 2 1946 2 2027 2 2126 2 2209 3 22
       2356 2 2440 1
239 0724 1 0835 2 0932 2 1000 1
239 1626 1 1729 2 1842 2 1927 2 2026 2 2109 3 2157 2 2256 2 23
```

```
MINIMUM MEALBREAK
46 50 51
50 42 44
51 44 42

PAID MEALBREAK
6 10 11
10 2 4
11 4 2
MINIMUM JOINUP
18 22 23
22 14 16
23 16 14
SIGNING ON & OFF
15 21 22
15 21 22
```

A.2 Duty Generation Parameter File

This file is used by IMPACS to control the formation of the large set of possible duties from which the schedule is selected. Some of the parameters are designed to prevent the formation of invalid duties. Others can be used to restrict the number of duties formed, by not allowing duties which are technically valid but unlikely to be useful; examples of this kind of parameter are the minimum stretch length and maximum mealbreak parameters. The values to be used are left to the scheduler's discretion.

The file contains lines describing the information required on the next line; as with the bus schedule data file, these must be present but are skipped over by IMPACS and simply serve as a reminder to the user.

The first parameter indicates whether a file of selected relief times has been set up or not (YES or NO, Y or N). "NO" means that all of the relief times will be available when the set of possible duties is formed. Normally, however, the relief time selection program will have been used and this line should read "YES".

The next three parameters are: the maximum break in a split duty; the maximum length of a joinup; and the minimum joinup. The minimum joinup value specified here is only required if for some reason the joinup should be longer than specified in the bus schedule data file: for instance, if the bus schedule data file contains only the travel time between pairs of points, but joinups must be a certain minimum length even if the travel time is zero.

The minimum length of a spell and the next parameter, which is the minimum length of a spell at the start or end of a running board, can be used to restrict the number of duties formed by IMPACS.

Some agreements specify a minimum length of, say, 3 hours for the break in a split duty. Otherwise, the minimum long break in a split duty can be left at 0, in which case the minimum break will be governed by the minimum mealbreak values in the bus schedule data file.

The minimum spreadover for a split duty is usually either explicitly stated in the agreement, or implied by the fact that split duties must have longer spreadovers than duties of other types.

The next section of the file gives various parameters to control the formation of different types of duty. Parameter values are required for five duty-types:

a) Early duties starting at the garage.

b) Late duties finishing at the garage.

c) Middle duties.

d) Spreadover or split duties.

e) Day duties.

The parameters which must be specified for each type of duty are listed below. Many of them will either have standard values or can be left at default values (0 or 3200). All values are given in hours and minutes with no punctuation (see example).

1. Minimum length of 1st stretch.
This requires the scheduler to specify the minimum time from the start of the duty to the start of the mealbreak (platform time + joinup, if the stretch works on two buses, but not including signing on or any of the allowances at the mealbreak). It is used to limit the number of duties formed by IMPACS.

2. Minimum length of 2nd stretch.
This refers similarly to the time from the end of the mealbreak to the end of the duty.

3. Maximum mealbreak.
The scheduler is asked to specify here the maximum length of mealbreak which could realistically occur.
The maximum mealbreak for split duties given here only applies to agreements which allow three-bus split duties to have two mealbreaks. This is then the maximum length of the secondary mealbreak. The maximum length of the main mealbreak in a split duty is specified earlier (on the fourth line of the file).

4. Earliest start of mealbreak.
5. Latest start of mealbreak.
6. Earliest finish of mealbreak.
7. Latest finish of mealbreak.
These four parameters can be used to define canteen opening and closing times, allowing for some leeway either side. For instance, suppose the canteen opens at 0700, and mealbreaks can start up to 15min. before that, provided that the canteen is open for at least 40min. of the mealbreak. Then the earliest start for a mealbreak is 0645, and the earliest finish is 0740. Otherwise, these parameters can sometimes be used to restrict the number of duties formed.

8. Minimum acceptable cost.
The cost is calculated by the WAGES routine, and unless its value falls between the minimum and maximum values specified here and in item 10, the duty is invalid. Most agreements do not specify a minimum cost, but it can be used to prevent IMPACS producing a lot of very short duties which are unlikely to be worthwhile considering.

9. Minimum cost for three-bus duties.
Since three-bus duties are usually relatively long, it may be possible to specify a higher minimum cost than for two-bus duties. This is useful because the number of potential three-bus duties is very large, and the number generated should be limited whenever it is reasonable to do so.
If three-bus duties of a particular type are not to be formed, this can be achieved by putting a minimum cost value greater than the maximum cost for that duty type.

10. Maximum cost.
This is often governed by the agreement.

11. Earliest signing-on time.
12. Earliest starting time on bus.
It would not normally be necessary to specify both of these for any duty type. The values are sometimes given in the agreement and sometimes

determined by practical considerations.
For early duties, both values can be 0; the earliest starting time for an early duty is automatically the time of the first bus out of the garage.
There is often no formal distinction between day and middle duties, but the parameters must be specified to avoid duplication (i.e. the same duty being formed twice, once as a day duty and once as a middle duty), for instance, by making the earliest starting time for middles later than the latest starting time for days.
It speeds up the formation of duties in IMPACS if earliest starting times can be specified for all types of duty, even if there is no rule concerning this in the agreement.

13. Latest starting time (on bus).

14. Earliest finishing time (off bus).
The earliest finishing time for late duties should be specified; this is the time that the first late bus returns to the garage.

15. Latest finishing time (off bus).
16. Latest signing off time.
As with items 11 and 12, it would not normally be necessary to specify both of these for any duty type.
The latest finishing time for middle duties is often the same as the latest finishing time for a mealbreak.
Latest signing off times or finishing times for early, day and split duties should be given, if possible, to speed up the generation of the set of duties.

17. Maximum length of stretch.
This means, as in (1), the time on bus + joinup, if any, and excluding allowances at the beginning and end of the stretch. If the agreement specifies a maximum stretch length including allowances, this is checked separately in the validation routine. The value given here is simply used to weed out some of the invalid duties at an early stage.

18. Maximum platform time.
As with maximum stretch length, this does not include any allowances.

19. Maximum spreadover.
The maximum spreadover is usually specified in the agreement. The simplest way of preventing IMPACS forming possible split duties for Saturday and Sunday schedules is to set the maximum spreadover to zero.

20. Minimum paid day.
The minimum payment for a duty if there is one, otherwise zero. The cost calculated by the WAGES routine, after checking against the minimum and maximum values given earlier, is increased to the minimum paid day if necessary.

Immediately after this section of the file, it is possible to list any duties which must be included in the schedule. The sample file does not include any prespecified duties; an example of this part of the file with one duty specified is as follows:

```
PRESPECIFIED DUTIES
1                       (Number of prespecified duties)
2 E                     (No. of spells of duty (1  3) & duty  type)
232  500  836          (1st spell: bus no.; time on; time off)
234  923  1127         (2nd spell: bus no.; time on; time off)
```

The duty type codes are E,L,M,S and D for early, late, middle, split and day duties respectively.
The program will not check a prespecified duty for validity.

The next item is the maximum number of mealbreaks allowed in a duty. This will be 1 or 2, depending on whether a three-bus duty must be composed of two stretches of work, so that one of the breaks must be counted as a joinup (even if longer than the minimum mealbreak) and the two spells of work either side of the joinup must together be short enough to form a valid stretch, or whether three-bus duties can be formed which have two mealbreaks.

The next item indicates whether the duty generation program is to look for a routine to form overtime pieces, or one-bus duties. Since many agreements do not allow work to be left out of the schedule to be covered as overtime, and also the form of permissible overtime pieces and one-bus duties varies considerably in those agreements which allow them, the duty generation program does not form any one-bus duties itself, but expects a separate routine to be provided if necessary. You should put Y if there is such a routine and it is to be used on this run and N otherwise (or YES or NO).

The next set of items defines periods of the day when joinups cannot start. This information is used to restrict the formation of three-bus duties, since the duty generation program can often produce an enormous number of such duties, even though very few may be required in the schedule. The start and finish time of each period should be specified, either all on one line, or using a separate line for each pair of times. Up to six periods can be defined. If there are no periods during which joinups need not be formed, the scheduler can put "NONE".

Example Of A Duty Generation Parameter File

```
SELECTION OF RELIEF TIMES USED?
YES
MAX-SPLIT-DUTY-BREAK MAX-JOINUP MIN-JOINUP
500                       50          0
MIN. LENGTH OF SPELL
100
MIN. LENGTH OF SPELL AT START OR END OF BUS
100
MIN. LONG BREAK IN SPLIT DUTY
40
MIN. SPREADOVER IN SPLIT DUTY
841
MIN LENGTH OF 1ST STRETCH
 130  100  130  130  130
MIN LENGTH OF 2ND STRETCH
 100  130  130  130  130
MAX MEALBREAK
 230  230  230  230  230
EARLIEST START OF MEALBREAK
0720 0000 0000 0720 0000
LATEST START OF MEALBREAK
3200 2108 3200 3200 3200
EARLIEST FINISH OF MEALBREAK
0810 0000 0000 0810 0000
LATEST FINISH OF MEALBREAK
3200 3200 3200 3200 3200
MIN COST
530 530 530 530 530
MIN COST (3-BUS DUTIES)
630 630 630 630 630
MAX COST
736 736 736 736 736
EARLIEST SIGNING ON TIME
```

```
0     0     0     0     0
EARLIEST START ON BUS
0     1200 1100 0     0720
LATEST START ON BUS
0724 1800 3200 1100 3200
EARLIEST FINISH OFF BUS
0     2340 1743 0     0
LATEST FINISH OFF BUS
3200 3200 2200 3200 3200
LATEST SIGNING OFF TIME
1800 3200 3200 2000 1800
MAXIMUM STRETCH LENGTH
 500   500   500   500   500
MAXIMUM PLATFORM TIME
 706   706   706   706   706
MAXIMUM  SPREADOVER
 840   840   840 1200   840
MINIMUM PAID DAY
0 0 0 0 0
PRESPECIFIED DUTIES
0
MAXIMUM NUMBER OF MEALBREAKS
1
OVERTIME PIECES TO BE FORMED?
NO
PERIODS WHEN JOINUPS CANNOT START
1530 1700
2130 3200
```

A.3 Relief Time Selection File

This file is required to run the relief time selection program.

For each bus which cannot be covered in a single spell, the program calculates pairs of relief times between which it will initially consider selecting times. This is done by dividing up the running board into bands of relief times approximately a spell length apart. The spell lengths to be used are specified in the data file. The values are given as hours and minutes, e.g. 330 = 3 hours, 30 minutes. The relief time selection program takes the values specified as the time on bus and does not include any allowances.

You are asked to specify two spell lengths (although they do not have to be different). One may be thought of as the maximum spell length and the other as the average spell length. This ensures that spells up to the maximum will be considered, but that when each running board is being divided up into bands of relief times, no running board will be cut up in such a way that all the spells on it must be as long as the maximum. For instance, if the longest spell on bus is 4:30 and there is a running board which could just be covered in three spells of work if they were all nearly 4:30 long, you may feel that in fact the maximum spell length you would expect to achieve most of the time is 4 hours and so this running board should be divided into four spells. Specifying an average of 4 hours will achieve this.

If the time worked on some buses is limited by the union agreement, this can also be specified. In the example shown below, the maximum time which can be spent on the 3 buses 103, 105 and 107 is 3 hours.

After this, the program will analyse the busgraph further, and attempt to form notional early and late duties using only the relief times already selected. Any relief times which do not appear to be useful at this

stage will be dropped from the selection.

For each early bus, the program attempts to form up to a specified number of early duties, the number being given in the data file. (10 seems to be a reasonable number to choose, as in the example). Late duties are formed similarly. If you answer YES to "ARE EXTRA RELIEF TIMES TO BE SELECTED IF NECESSARY?", relief times not initially selected can be considered if this seems to be necessary to form a good set of duties.

The first part of the data file specifies periods of the day when relief times should not be selected. The banned periods can be of two types: type 1 periods, when relief times should be avoided if possible (e.g. during the morning or evening peak) and type 2 periods when relief times must not be chosen (e.g. after the canteen closing time in the evening). For each banned period, two lines of data should be given: the first has the type, 1 or 2, and the second the limits of the period. This section of the file is terminated by a zero.

If you answer NO to the next question, "RELIEF TIMES CAN BE CLOSE TOGETHER?", the program will try to avoid choosing two relief times less than 20 minutes apart.

Example Of A Relief Time Selection Data File

```
BANNED PERIODS
2
0 0600
2
2144 3200
0
RELIEF TIMES CAN BE CLOSE TOGETHER?
N
MAXIMUM STRETCH LENGTH (ON BUS)
430
EXPECTED AVERAGE STRETCH LENGTH (ON BUS)
400
IF TIME ON SOME BUSES IS LIMITED, MAX. TIME ALLOWED & BUS NUMBI
300
3 103 105 107
NUMBER OF 2ND STRETCHES OF EARLY DUTY, ETC. TO BE KEPT
10
ARE EXTRA RELIEF TIMES TO BE CHOSEN IF NECESSARY?
Y
```

A.4 ZIP Parameter File

The ZIP program requires a final data file which controls the way in which the program is run.

The first line specifies, via the START parameter, which mode the program is to be run in, and whether or not penalty costs are to be calculated for each duty and added to the total cost. The START parameter has a value between 0 and 4: the meaning of these values is described below.

START=0. ZIP starts from scratch, finding an intermediate solution (in which the duties may have fractional values) and then going on to find a solution in which the duties have integral values, forming a practicable schedule which can be printed immediately.

START=1. ZIP similarly starts from scratch and produces an intermediate solution, but then stops.

START=2. ZIP picks up an earlier intermediate solution, finds a new intermediate solution if necessary (i.e. if the costs of the duties have been changed, or new constraints have been added as described below so that the old intermediate solution is no longer the best), and then goes on to find a solution with integral values.

START=3. Similarly, ZIP picks up an earlier intermediate solution, finds a new one if necessary, but then stops.

With each of the above values of the START parameter, the ADD-PENALTIES parameter is required to control whether or not penalty costs are included in the cost of a duty. Y or YES means penalties are added, N or NO means they are not.

START=4. This mode should be used if penalty costs are to be included in the total cost of duties and the ZIP program is required to find an integer solution without any intervention from the user. The ADD-PENALTIES parameter is taken care of automatically, but instead the scheduler should specify either a target number of duties, or zero if the number of duties the schedule is expected to contain is unknown.

If a target number of duties is specified, this mode will either produce a schedule with this number of duties, or print a message to say that it is impossible to achieve the target number of duties from the generated set of duties. If no target number of duties is given, this mode is designed to ensure that the solution will have the minimum number of duties. A special mode is necessary because penalty costs may mean that the cheapest solution is not the one with fewest duties.

If a target of n duties is specified, the program will automatically add a constraint "TOTAL GE n". This is because it saves computer time to restrict the program in this way, rather than to allow it to produce an intermediate solution with slightly under n duties (say 39.5 instead of 40) and then to increase the total to n. However, if it is in fact possible to find a schedule with fewer than n duties, the ZIP program will be prevented from looking for one. A target should only be specified if the scheduler is confident that it is accurate, or wishes to prevent a schedule with fewer duties being formed.

The next set of data in the file consists of a number of cost parameters. The first parameter is the constant cost added to the other cost components of each duty. The other costs, calculated by the WAGES routine, and the penalty cost routine if used, are normally calculated in minutes and are then added together. The purpose of the additional large constant is to ensure that the first priority in the ZIP program is to minimise the number of duties, and only secondly to minimise the total of the other costs. It should be sufficiently large that the ZIP program cannot reduce the total cost of the solution by substituting two cheap duties for one expensive duty, and usually 2000 is a reasonable value to choose.

(If the costs of duties include penalty costs, a few duties may incur several different penalties and so work out extremely expensive. Effectively, this means that these duties should never be used unless the work can be covered in no other way, and it would be acceptable in this case if the constant additional cost allowed ZIP to cover the same work by using two cheaper duties.)

The second cost parameter is the cost of over-cover, per minute. The value of this can be varied to reflect the importance of producing a schedule without over-cover, but a value of 3.0 has usually proved satisfactory in the past. This means, for instance, that if a piece of work 20 minutes long were covered by two duties, the total cost of the solution would be increased by 60. (If the same piece is covered by three duties, the cost would be increased by 120, and so on.)

Usually, a value of 3.0 will be sufficient to deter ZIP from producing a schedule with over-cover. If the schedule is inherently liable to over-

cover, e.g. because the agreement does not allow split duties to be used to cover the peaks, and many duties in the manual schedules contain periods of stand-by, ZIP will probably produce a schedule with over-cover. If this is unacceptable, it may be possible to get rid of all or most of the over-cover by increasing the cost parameter. Often, however, it is preferable to remove the over-cover by editing the duties manually and so simulate stand-by periods.

The remaining cost parameter defines the cost of an uncovered piece of work. As far as the ZIP program is concerned, a duty schedule in which every piece of work is uncovered is actually feasible, but it is prevented from offering this as a solution if leaving the work uncovered is more expensive than assigning a set of duties. The parameter is the basic cost of an uncovered piece of work and should be given a large value, as in the sample file. The cost of an uncovered piece of work will be this value plus the length of the piece in minutes.

At the end of the file, any extra constraints can be listed. These are limitations on the number of duties of a particular type, or on the total number of duties, or prevent the ZIP program from leaving work uncovered.
The possibilities are:

EARLIES LATES MIDDLES SPLITS DAYS OVERTIME TOTAL JOINUPS	GE LE	*number of duties*

or

NO UNCOVERED PIECES

where GE stands for "greater than or equal to" and LE for "less than or equal to". For example, the constraint MIDDLES LE 3 would mean that the schedule is not allowed to contain more than 3 middle duties. JOINUPS LE 2 would mean that the schedule could contain at most 2 three-bus duties.

If overtime pieces or duties have been formed, the number of these can be constrained in the same way as other types of duty. The total number of duties, in a constraint such as TOTAL LE 50, also includes the overtime pieces. The total does not, however, include uncovered pieces of work.

IMPACS is allowed to leave work uncovered, but the cost of doing so is normally too high for this to happen unless the program can find no other way of constructing a schedule, for instance if the set of duties which it is selecting from has no duty covering a particular piece of work, or if the extra constraints make it impossible to construct a schedule without leaving work uncovered. It may sometimes be necessary to add the "NO UNCOVERED PIECES" constraint when modifying an existing solution, for instance to establish the minimum number of splits required in a schedule, since if work can be left uncovered, split duties can always be completely eliminated.

Example Of A ZIP Data File

```
START  ADD-PENALTIES OR TARGET NUMBER OF DUTIES
1              Y
COST-OF-DUTY COST-OF-OVER-COVER COST-OF-UNCOVERED-PIECE
2000                  3.0                    3000
EXTRA CONSTRAINTS
SPLITS LE 3
```
