Scheduling, Timetabling and Rostering - A Special Relationship?

Anthony Wren
Professor of Scheduling and Constraint Management
School of Computer Studies, University of Leeds, Leeds LS2 9JT
wren@scs.leeds.ac.uk

Abstract

Computer solution of timetabling, scheduling and rostering problems has been addressed in the literature since the 1950's. Early mathematical formulations proved impossible to solve given the limited computer power of the era. However, heuristics, often very specialised, were used for certain problems from a very early date, although the term *heuristic* was not generally recognised until later; a few guaranteed optimality, some consistently produced good solutions, but most became unwieldy when adjusted to deal with practical situations. In some cases, weaknesses in the heuristics were overcome by appeal to manual intervention. Mathematical approaches to some problems returned to favour, successfully, around 1980. Some of the subsequent developments of these are very powerful in practical situations, but they are no panacea, and metaheuristics are the flavour of the nineties.

This paper explores the relationships between the problem types, and traces the above developments as applied principally in the areas of Vehicle Routeing and Scheduling, Driver Scheduling, Job Shop Scheduling and Personnel Rostering. Parallels are drawn with Class and Examination Timetabling, but these subjects themselves are not examined, as they are covered extensively elsewhere in this volume.

1. Introduction

It is tempting in presenting a paper with the present title to start by setting up some definitions so that we may distinguish the three terms of the title. However, a browse through relevant literature reveals very different uses of the terms. We shall therefore start by surveying some of these uses, and then introduce definitions which we shall use within the current paper.

We shall then trace some developments of scheduling, timetabling and rostering as defined below, drawing to a great extent from the author's own principal areas of experience, but introducing parallels from a wider domain.

We start with the most widely used term: scheduling. What is Scheduling?

Authors of works on scheduling have tended to pre-empt the term to their own domain:

> The Vehicle Scheduling Problem (VSP), discussed in OR since the 1960's, refers only to the construction of routes for commercial vehicles, as if these were the only vehicles to be scheduled. This is now often called the Vehicle Routeing Problem (VRP);
>
> A particular range of literature [1-5] implicitly refers to buses, trains and their drivers when discussing scheduling;
>
> French's text on Sequencing and Scheduling [6] discusses nothing that moves;
>
> Most OR/AI people mean some form of production scheduling when they use the term "scheduling";
>
> Some literature on class and examination timetabling refers to the process of drawing up the timetable as scheduling.

Scheduling is ...

> Definition of a pattern among objects or resources so as to satisfy certain criteria?
>
> Ordering or sequencing objects in order best to satisfy certain objectives within certain restrictions?
>
> Allocation, subject to constraints, of resources to objects being placed in space-time?

We shall consider the objective of scheduling to be:

> to solve practical problems relating to the allocation, subject to constraints, of resources to objects being placed in space-time, using or developing whatever tools may be appropriate. The problems will often relate to the satisfaction of certain objectives.

Although some may consider scheduling and timetabling to be separate activities, we shall use the term *scheduling* both as a generic term and to cover specific types of problem, and shall consider timetabling, sequencing and rostering as special cases of the generic scheduling activity.

Scheduling may be seen as the arrangement of objects into a pattern in time or space in such a way that some goals are achieved, or nearly achieved, and that constraints on the way the objects may be arranged are satisfied, or nearly satisfied.

The objects may be people, vehicles, classes, examinations, machines, jobs in a factory, etc. Often the formation of objects may be seen as part of the scheduling process. For example, items of work may be arranged into personnel shifts which have to be grouped into a roster. The formation of shifts may be seen as a specific scheduling process within the larger process.

The pattern may be an ordering of events, a set of legal shifts defined in terms of work to be done during the shifts, a structure of routes, a matrix of allocations, etc.; the overall pattern may have to be created as part of the scheduling process, or may pre-exist as a template which is characteristic of the problem being tackled.

The constraints define physical or legal relationships among the objects or between the objects, other objects and the pattern. They govern the ways in which objects may be fitted together or into the pattern. Constraints may be seen as rules which hinder the achievement of goals. However, they may also be seen as part of the problem specification which may be used to guide the solver towards a solution. Some constraints may exist only in the eye of the problem owner, and it may be part of the solution process to indicate the extent to which a solution might be improved if a constraint or constraints were to be relaxed, so that the owner might decide whether the constraint is necessary or how much it might be worth spending to abolish the constraint.

Sometimes the words schedule, sequence and timetable are loosely used as if they were synonymous. Here we shall make some distinctions, and although we may sometimes have to overstep our definitions in order to conform to accepted practice, we shall try to be consistent as far as possible.

A *timetable* shows when particular events are to take place. It does not necessarily imply an allocation of resources. Thus a published bus or train timetable shows when journeys are to be made on a particular route or routes. It does not tell us which vehicles or drivers are to be assigned to particular journeys. The allocation of vehicles and drivers is part of the scheduling process. Although timetabling is strictly the design of the pattern of journeys, this pattern may be devised as part of a process which bears in mind whether it is likely that an efficient schedule may be fitted to the resulting journey pattern.

In the rail domain, the term timetabling is often used to refer to the construction of a path (with times) for a train through a system. Thus, the Flying Scotsman used to leave Edinburgh (Waverley) at 10 a.m.; the job of timetabling was the process of finding times and paths through the rail system which did not conflict with other traffic (revising such other traffic as necessary) until it reached Kings Cross at 6.30

p.m. (in 1922) or 6.05 p.m. (1948). This form of timetabling has been the subject of some research, but little, if any, practical implementation over any but the simplest systems. Carey [7] has recently reported on timetabling and track allocation in large stations.

A class timetable also shows when particular events are to take place. In certain circumstances there may be no scheduling activity necessary in drawing up such a timetable. In an infants' school where a single teacher is responsible for all the activities of a particular class, and where these activities all take place in the same room, a timetable is nothing more than a statement as to the times at which particular activities will take place (as is the public train timetable above). By contrast, a university examination timetable will normally include room assignments drawn up in the knowledge of group sizes and of special facilities needed. It will have been devised subject to hard or soft constraints, for example on the numbers of examinations to be taken by individual students in consecutive periods. A university class timetable has also to take into account the availability of individual lecturers. The activities of drawing up examination and university class timetables may be considered as scheduling activities.

A *sequence* is simply an order in which activities are carried out. For example, the order in which jobs are processed through the machines of a factory, if jobs pass through each machine in the same order, is a sequence. Sequencing may take into account costs related to one particular job being followed by another (machine conversion costs etc.). The problem of sequencing jobs in these circumstances is known as a flow shop problem.

A *schedule* will normally include all the spacial and temporal information necessary for a process to be carried out. This will include times at which activities are to take place, statements as to which resources will be assigned where, and workplans for individual personnel or machines.

We have already tentatively defined scheduling as allocation, subject to constraints, of resources to objects being placed in space-time. Thus scheduling is the process of forming the schedule, including assignment of resources. We note that French [6] uses the term *timetabling* for this process, and other authors will use other conventions.

2. A Simple Sequencing Example

We shall introduce here a simple example of a sequencing problem, showing some common types of solution heuristic.

A travelling salesman problem (TSP) in which cities have to be visited in the order which minimises the total distance travelled or the total time elapsed is an example of a sequencing problem, as is the more general vehicle scheduling problem (VSP), nowadays more usually and properly called the vehicle routeing problem (VRP), where routes have to be constructed from one or more depots to delivery outlets subject to limits on vehicle capacity and on route distance or duration. The solutions to such problems may be expressed as sequences of points, although depots may each have to appear several times in a sequence. Where further constraints exist on times when certain points are visited, on mixtures of goods which may be carried, or on sizes of trucks which may serve particular points, we may consider ourselves to have a true scheduling problem.

In fact, the TSP and a simple job sequencing problem may be considered as logically equivalent, and we shall shortly introduce a particular example of these. Just as the TSP is a simple example of a range of routeing problems, so the single machine problem is a simple example of a range of job scheduling problems. In both cases we may gain an insight which may be useful in the more complex problems by studying these simple ones.

Consider the processing of nine jobs passing through a machine. The machine may have to be adjusted or cleaned between operations, i.e. between pairs of jobs. There will be costs associated with the adjustment or cleaning. These costs may be combinations of time delays and actual costs of materials used in a conversion process, and we shall want to find the sequence of jobs which minimises the total of these; the cost of job j following job i (c_{ij}) will be the conversion cost of the machine from processing i to processing j. Let us assume that there are costs c_{0j} associated with setting up the machine for job j and costs c_{i0} for clearing the machine after job i. In practice such costs are often ignored and can be treated as zero in the table, but we shall assume non-zero costs here. Diagonal elements of the table are infinite, since a job cannot follow itself.

	J0	J1	J2	J3	J4	J5	J6	J7	J8	J9
J0	X	35	33	11	20	17	36	45	25	15
J1	33	X	50	44	22	26	21	43	38	46
J2	36	47	X	40	48	27	34	25	13	21
J3	12	46	39	X	26	28	47	56	34	19
J4	22	25	45	27	X	24	34	52	37	35
J5	16	27	30	25	25	X	19	29	14	21
J6	33	22	38	44	31	22	X	22	24	39
J7	48	42	26	59	51	26	22	X	22	40
J8	22	35	13	33	38	15	23	20	X	18

J9	13	45	23	17	33	20	41	41	17	X

Let us now consider a trial solution in which the jobs are processed in an arbitrary, here numerical, order. This solution, including setting up and clearing may be written:

```
J0-J1-J2-J3-J4-J5-J6-J7-J8-J9-J0
```

with cost:

```
35+50+40+26+24+19+22+22+18+13 = 269
```

There is no reason to think that this is a good solution, but we shall consider this question later. We have simply ordered the jobs into an arbitrary sequence and determined the cost of the result.

Now consider a travelling salesman based at city J0 who has to visit nine other cities, J1 - J9, and return to base. Distances d_{ij} between cities are given as in the table above. We require to find the route (or sequence of cities visited) which minimises the total distance travelled. (Since the route returns to base it doesn't really matter which we consider as the starting point, but for comparison with the flow shop problem, we have chosen city J0). The sequence J0, J1, J2, J3, J4, J5, J6, J7, J8, J9, J0 represents a solution:

```
J0-J1-J2-J3-J4-J5-J6-J7-J8-J9-J0
```

with distance:

```
35+50+40+26+24+19+22+22+18+13 = 269
```

Suppose now we have a class timetabling problem in which a cohort of students attends nine classes in a day. There may be notional costs associated with moving between classes, as shown in the above table, and with starting the day with any class or finishing with any class. The sequence J1, J2, J3, J4, J5, J6, J7, J8, J9 represents a solution, but we may include coming in the morning and going home at night in the representation:

```
J0-J1-J2-J3-J4-J5-J6-J7-J8-J9-J0
```

with cost:

```
35+50+40+26+24+19+22+22+18+13 = 269
```

The timetabling of this particular cohort of students may be just part of a larger timetabling problem, but given a tentative solution to the overall problem, it may be possible to make piecewise improvements to that solution by revising the above sequence (which may involve reallocating the calls on teachers or rooms).

More briefly, we may represent the given arbitrary solutions to any of these problems as:

```
 0   1   2   3   4   5   6   7   8   9, with cost:
  35  50  40  26  24  19  22  22  18  13  =  269.
```

Commonly scheduling problems are treated by heuristics, for example we may seek to improve the solution by switching a pair of jobs, cities or classes in the sequence, e.g. 2 and 5, to obtain the new sequence:

```
 0   1   5   3   4   2   6   7   8   9, with cost:
  35  26  25  26  45  34  22  22  18  13  =  266.
```

If this does not violate any constraints, we have achieved an improvement, although other swaps might have produced a worsening. The heuristic given above, if continued by trying every pair of swaps iteratively until no improvement can be found, is an example of a very common class of heuristics leading (usually) to a local optimum. It may be applied to very many timetabling and scheduling problems. If swaps are processed in the logical sequence 1-2, 1-3 , ... 1-9, 2-3, 2-4 etc., repeating after trying 8-9, one obtains the solution:

```
 0   3   4   1   5   6   7   8   2   9, with cost
  11  26  25  26  19  22  22  13  21  13  =  198
```

The solution is known as 2-optimal; no pairwise change will produce an improvement. The quality of this (possibly) local optimum in a general problem type usually depends on the starting solution, although in one problem type known to the author [8-11] a similar process has in many thousand problem instances always led to solutions believed to be optimal.

There is no way in general of knowing whether the solution is optimal, although in this particular case a better solution is known (obtained from the same first solution by considering moving each object in turn to its best position in the sequence):

```
 0   4   1   6   7   5   8   2   9   3, with cost
  20  25  21  22  26  14  13  21  17  12  =  191
```

Although this better solution is the best that can be achieved by the particular heuristic, given the starting solution, it is not necessarily optimal. The reader may wish to check whether it is 2-optimal.

Better solutions may often be reached by considering changing the positions of triples of objects, but this requires much more computer time which may not be justified (there are six ways of arranging triples, all of which may have to be evaluated, but only two ways of arranging pairs). Some n-optimal processes depend on the links between successive objects rather than on the sequencing of the objects. Pairs of links may be arranged in three ways (AB, CD; AC, BD; AD, BC), but triples of links may be arranged in fifteen ways; often many of these ways are infeasible.

3. Definitions Revisited

We have already defined the goal of scheduling in its broadest sense as:

> to solve practical problems relating to the allocation, subject to constraints, of resources to objects being placed in space-time, using or developing whatever tools may be appropriate. The problems will often relate to the satisfaction of certain objectives.

The activities known as *timetabling, rostering* and *sequencing* all conform to the above definition. However, we shall use the following more restrictive definitions in the remainder of this paper:

> *Scheduling* is the allocation, subject to constraints, of resources to objects being placed in space-time, in such a way as to minimise the total cost of some set of the resources used. Common examples are transport scheduling or delivery vehicle routeing which seek to minimise the numbers of vehicles or drivers and within that minimum to minimise the total cost, and job shop scheduling which may seek to minimise the number of time periods used, or some physical resource.
>
> *Timetabling* is the allocation, subject to constraints, of given resources to objects being placed in space-time, in such a way as to satisfy as nearly as possible a set of desirable objectives. Examples are class and examination timetabling and some forms of personnel allocation, for example manning of toll booths subject to a given number of personnel.
>
> *Sequencing* is the construction, subject to constraints, of an order in which activities are to be carried out or objects are to be placed in some representation of a solution. Examples are flow-shop scheduling and the travelling salesman problem.
>
> *Rostering* is the placing, subject to constraints, of resources into slots in a pattern. One may seek to minimise some objective, or simply to obtain a feasible allocation. Often the resources will rotate through a roster.

Some problems may fit more than one of the above definitions, and the terms tend to be used rather loosely in the workplace and in the scheduling community.

In some of the above we have referred to satisfying or to minimising. It should be remarked that many of the problems which we are treating do not have a well-defined objective. We may sometimes justify the use of ameliorating or non-optimising methods partly because different players will have different views of the objective, but in reality such methods are often used simply because no optimising (or exact) method is practicable.

4. Scheduling Problems Reviewed

The author first formally met scheduling problems in 1961 when he was assigned to a rail locomotive scheduling exercise for which he subsequently assumed full responsibility and designed simple heuristics. This led in 1963 to the implementation of what was probably the world's first computer-produced rail locomotive schedule. In 1966 he turned to delivery vehicle routeing, and with a student developed heuristics which were used in practical exercises with a number of clients over the next few years.

In 1967 he was approached by the bus industry, for whom he and his group developed a wide range of scheduling programs over the next twenty-eight years. The earlier programs, until the end of the 1970's, were based on heuristics. Although these heuristics were designed independently, they can be classified according to the scheme of Müller-Merbach [15]. Some of these (for bus scheduling) are very robust and still in commercial use. In another problem, however, the heuristics tended to work only in cases very similar to those for which they were designed.

The 1960's also saw the start of research into computer methods for many other scheduling, sequencing, timetabling and rostering problems. Some of these problems will be introduced here, referring mainly to early research. In many fields the first approach of the researchers, most of whom came from mathematics or physical sciences, was to design a mathematical model. However, practical implementations of the models were impossible given the computer power then available. The first serious solution attempts therefore almost always involved specially designed heuristics, although the term *heuristic* was not yet generally known.

In almost all cases this work was followed through the 1970's and 1980's by further heuristic developments, coupled in some cases with integer linear programming. By the 1990's a variety of recently discovered or rediscovered approaches was being enthusiastically applied to most scheduling problems, often, perhaps regrettably, driven more by a desire to use the approach than by a need to solve the problem (see Section 5).

4.1 Job Shop Scheduling

Almost certainly the most frequently discussed form of scheduling in the OR/AI literature, job shop scheduling concerns the processing of jobs through a number of machines, subject to limits on various resources. A number of objectives have been identified, for example, minimising the makespan (the total time to completion of all jobs), the total lateness of jobs, or the number of late jobs. A simple form of job shop scheduling is flow shop scheduling, in which jobs move through each machine in the same order; flow shop scheduling is essentially a sequencing problem.

Most job shop scheduling problems are very specialised and are solved to local optima by special heuristics. No optimising method is known for the general problem of more than two machines. However, even before the 1960's, Johnson [12] had developed an optimal strategy for a two-machine problem. Johnson's work is a rare example of a heuristic which can be proved to produce the optimal result, as are the later heuristics of Moore [13] and Lawler [14] for single machines. Johnson's paper was published in the first volume (1954) of the Naval Research Logistics Quarterly which was for many years a standard source of scheduling papers.

4.2 Travelling Salesman Problem

The TSP has already been introduced. It is widely studied, perhaps more as a theoretical exercise than in order to solve practical problems. However, many problems can be reduced to TSP's (see above), so that solution methods for TSP's can be adapted for them, while vehicle routeing problems may be seen as extensions of TSP's. Müller-Merbach [15] used the TSP to illustrate a wide range of heuristics which he classified as either First Feasible Solution or Iteratively Improving, with sub-classifications. This seminal paper was perhaps the first to try to bring together a theory of heuristics which previously had been introduced as diverse ad hoc, often trial and error, methods.

One of the most successful approaches to the TSP in the early era was due to Lin [16], using an n-optimal method.

4.3 Vehicle Routeing Problem

The delivery vehicle routeing problem attracted much attention in the operational research community during the late 1960's and early 1970's. Commonly (and arrogantly) known within OR as *the* vehicle scheduling problem (VSP), it was originally formulated as the problem of designing routes for delivery vehicles from a single depot subject to constraints of load and distance. Although Dantzig and Ramser [17] postulated a solution based on linear programming, the first practical method was probably the heuristic of Clarke and Wright [18] which built up a first feasible solution and was successfully implemented, first for the Co-operative Wholesale Society based in Manchester. The method formed the basis of several commercial routeing packages, although it was already known to the research community to produce poor solutions; these were nonetheless often much better than existing manual solutions. Gaskell [19] experimented with several variations of the

Clarke and Wright method, introducing six test data sets which have since been widely used.

Christofides and Eilon [20] introduced a 3-optimal method for the VRP, building on Lin's work on the TSP. This applied the approach to several randomly generated starting solutions. They applied their method to the six problems of Clarke and Wright, and to three further ones, showing that it was significantly better than any known variants of Clarke and Wright.

The author, initially with a student [21, 22], designed a multi-depot system which out-performed the above single depot systems on the nine data sets referred to above, and which was used to assist in strategic decision making (e.g., depot siting, creating routes to be operated regularly over a long time period) by Phillips Industries and by Christian Salvesson for deliveries to Marks and Spencer.

The heuristics of Wren and Holliday [22] commenced by building up routes, considering delivery points in order of directional angle (orientation) from the depot. Points were inserted in the cheapest position in an existing route where possible, and otherwise a new route was begun. Where the number of vehicles was limited some points might remain unserved. The search was repeated using a number of different initial orientations, and the cheapest set of resulting routes initiated a set of refining heuristics. (The same approach was later proposed by Gillett and Miller [23], who introduced it as the *sweep algorithm*, without the refinement stages.)

Wren and Holliday used a number of refining heuristics, principally with the goal of reducing costs; some attempted to reduce the number of routes used, and others sought to fit in points that were omitted initially, replacing others of lower priority. The cost reduction routines each performed a different action: moving points from one route to another; exchanging points between two routes; removing crossovers from routes; rebuilding routes according to the initial route construction method applied to a single route or to two adjacently oriented routes which may have become tangled, and keeping the new route structures only if they were better than the old.

These refining heuristics each searched all points or routes until no improvement remained, and were called cyclically in turn until no improvement resulted from a complete cycle, or until a certain time limit was reached (computer time was severely rationed in 1970!). Thus each call of each heuristic found a local optimum. Moving on to another heuristic corresponded to changing the shape of the search space. Overall, the solutions achieved were very good, and generally out-performed Christofides and Eilon on the nine test problems. As in many applications of heuristics, best results were achieved when cost-neutral changes were allowed, although this implied that each heuristic should be continued through several non-improving cycles before being terminated.

Although further mathematical programming formulations of the VRP were produced, these were generally too complicated for solution (but see [24] for an approach using heuristics to reduce the size of a problem which was then solved using integer linear programming). Laporte [25], and others earlier, have surveyed the development of computer methods for the VRP.

Although many more attempts were made to find good algorithms for the VRP, complicating factors such as many depots, different vehicle types, time windows for certain deliveries, etc., inhibited their application to practical problems, and it became evident through the 1970's that fully automatic systems were seldom suitable for solution of tactical problems. For this reason, many commercial systems were developed which used some (often primitive) form of heuristic to build an initial solution, but expected the user to adjust this interactively, using the computer to check out possibilities.

4.4 Timetabling

As timetabling is the theme of this volume, it is unnecessary in this paper to review in detail the large body of research devoted to this subject. Most timetabling research has been undertaken on problems in educational institutions, partly because these were real problems in the researchers' organisations, and partly because of their interesting complexity.

The rail timetabling problem has already been introduced and is the subject of continuing research by Carey (e.g. [7]). Other timetabling problems in the transport areas have been tackled by Kwan [26] and Keudel [27] among others.

Kwan introduced a method for determining suitable timings for regular bus journeys along common stretches of route where any individual service might share different common stretches with different groups of services. The objectives were to produce an acceptable solution to the problems of minimising the number of vehicles and of achieving even intervals between the different services on the common stretches. The method used the computer to present possible solutions to the user, allowing the goals to be changed interactively.

Keudel built on a method of Günther [28] to develop a system which designed timetables for intersecting bus services so as to achieve a good compromise between the number of buses and the waiting times of passengers making interchanges.

One of the earliest educational timetabling systems (for examinations) was developed in the University of Leicester by Cole [29] and published in 1964, while another by Wood [30] four years later took advantage of the greater power of the Atlas computer. Both these authors, and many later ones, used strategies that might now come under the constraint satisfaction banner. Wood used real 1967 data from the

University of Manchester to schedule 5730 students taking 1323 papers into between 24 and 30 periods according to a range of alternative constraints; each schedule took 10 minutes to produce, which seems remarkable for that era. Wood's strategy was basically to schedule the subject that needed the largest room first, but much of his paper is taken up with the technicalities of the binary representation needed to fit the information into what was then the world's largest computer.

If one reads the papers of Cole, Wood and their many successors, one finds that most claim success for their methods, yet none have survived in their institutions for more than a few years, if that. Wood quotes the University of Manchester authorities as saying in 1967 that "the existing system is stretched to its limits", yet most, possibly all, British universities (even the larger ones) struggled on using manual examination and class timetabling into the 1990's until pressures of modularisation forced proper consideration of automatic methods. A major reason for the lack of proper implementation of earlier methods appears to have been the distrust of university administrators of anything emanating from their academic colleagues.

An unusual timetabling problem tackled by Parker, Parker and Proll [31] will be mentioned here as an indication that serious timetabling problems arise elsewhere than universities. This related to the scheduling of interviews during parent-teacher evenings in a high school; a single evening concerned one year's pupils and might be attended by approximately forty teachers and 150 parents. Commitments of both teachers and parents might prevent either from being present during the whole evening. The method commenced with a heuristic which produced a feasible schedule, this being followed by optimisation of each teacher's schedule in turn while the remainder of the schedule was fixed, the goals of the optimisation taking into account parents' as well as teachers' goals. This treatment was possible because gaps in the parents' schedules allowed significant restructuring. The program was used successfully in Leeds schools for several years until with the advent of PC's commercial programs became available which linked some kind of scheduler with a school database.

More recent advances in class and examination timetabling are discussed elsewhere in this volume, and are therefore not considered further here, although timetabling problems are noted in some of the sections below in relation to the techniques outlined.

4.5 Rostering

Rostering problems arise in a large number of situations. Once shifts have been produced showing the daily work of personnel, these shifts are placed into a roster to show which shifts are worked by individuals on particular days. Usually the pattern of shifts for an individual in any week (or over a longer period of time) has to conform to particular rules. Rostering research has covered a wide range of occupations; policemen, nurses, bus and train drivers, etc.

We mention here bus driver rostering, where the daily shifts are grouped into rows of weeks or fortnights according to given hard or soft rules. These rules govern such factors as the positioning of rest days in the roster, grouping of shifts of particular types into certain rows, the observance of minimum times between the end of one day's work and the beginning of the next, and the maintenance of similar starting times on successive days. Drivers then rotate through cycles of rows. The cost depends on the extent to which individual rows exceed a target amount of work, so that an objective is to even out the amount of work in each row while minimising the violation of soft constraints. Bennett and Potts [32] in Adelaide tackled this in 1968 by considering a matrix in which rows represented fortnights of work (early week followed by late week) and the elements identified the shifts to be worked on the appropriate days. A two stage process was developed.

The first stage developed a pattern of rest days such that the correct number of working days could be fitted into each of the fourteen columns. The objective was to maximise first the number of four-day weekends, then three-day, then two-day weekends, then consecutive days off during the week. An initial solution was fitted to this and refined by a heuristic which kept thirteen columns fixed and reassigned the elements of the remaining column optimally using the Hungarian algorithm. Each column was reassigned in turn and the process was repeated until no improvement was produced. This was similar to the approach of Parker, Parker and Proll [31] to the scheduling of parent-teacher evenings.

The author and colleagues adapted the process of Bennett and Potts for British bus driver problems, replacing the Hungarian algorithm by a heuristic which simply exchanged elements in the column being considered. This produced satisfactory solutions in terms of the specified constraints relating to the placing of shifts of particular types in particular positions of the roster. Unfortunately, the bus company for which this work was done frequently violated their own constraints, and following a change of personnel in the company it proved impossible to determine the circumstances in which such apparent violations might be acceptable. However, later work with a firm of consultants enabled the author to develop a partly interactive bus driver rostering system based on this work; this is now being used by many bus operators.

The author has also worked with a student [Townsend, 33] to develop a rostering system for London Transport Buses. This was a very specialised problem in which all shifts were paid for eight hours; the goal was to devise suitable positions in the roster for rest days and for particular types of shift, and then to fit the shifts in such a way that shifts on consecutive says had similar starting times. Unfortunately, the rostering rules in London were changed dramatically before the system could be implemented.

More recently, colleagues [Smith and Bennett, 34] have used constraint satisfaction techniques to produce a system to develop rosters for anaesthetists in a major Leeds hospital.

Although the rules are in general very different, rostering is perhaps closer to examination and class timetabling than is any other problem area treated here. The final result is a matrix of allocations formed to take account of the relationships between the objects allocated to the slots. Some of the processes developed by Bennett and Potts can be adapted to ensure that gaps in teaching loads or in student commitment are spread equitably through a timetable matrix, while both the Bennett and Potts use of the Hungarian algorithm and the author's improving heuristics can improve a first attempt at a timetable.

5. Review of Modern Methods

Other work in scheduling is far too extensive to review here, but we shall introduce some of the current methods, referring to a few applications, before introducing the author's own work on a number of transport scheduling problems.

Simulated annealing is analogous to the annealing of materials to produce sound low energy states (e.g., large crystals). The basic annealing model was developed in 1953 by Metropolis et al. [35], while simulated annealing as an approach to optimisation problems was developed by Kirkpatrick and colleagues in the early 1980's [e.g., 36], although claims have been made for its earlier discovery.

Simulated annealing addresses the problem of escaping from local optima by adopting a process controlled by a parameter known as the temperature. A heuristic designed for the problem in hand defines the neighbourhood of an initial solution, and a possible move within that neighbourhood is generated at random. The move is accepted if it produces a better solution; a worsening move is accepted with a probability governed by the cost of the move and the value of the temperature parameter. At a high temperature a given worsening move is more likely to be accepted, leading possibly to an escape beyond the contours of a local optimum. As the process continues, the temperature is reduced until at a very low temperature only non-worsening moves are accepted. The process is analogous to physical annealing in which at high temperatures the molecules are moving rapidly and may bring the material temporarily into a higher energy state; as the material cools, the molecules slow down until they subside into a smooth low-energy state; the energy is analogous to the function being optimised.

The success of simulated annealing depends on the starting temperature, the heuristic which gives the initial solution, the refining heuristic, and the cooling schedule (the rate of temperature decrease). The reasons for the success of simulated annealing are not generally well understood, but theoretical work by Sorkin for a PhD thesis has led to some interesting results [37].

A bibliography on simulated annealing was published in 1988 by Collins, Eglese and Golden [38]. Recent timetabling work using simulated annealing includes

Thompson and Dowsland [39] and Ross and Corne [40]. Osman [41] discusses simulated annealing with reference to the vehicle routeing problem.

Tabu search, due to Glover [42, 43], escapes from a current solution by applying a greedy heuristic, moving to the best neighbouring point. If the new solution is worse it is accepted (unless it has previously been defined as tabu). The search is prevented from falling back into local optima by a procedure which makes certain moves tabu if they would bring the search to a position similar to one recently visited. Tabu moves are accepted if they lead to a better solution than has previously been found, and certain steps may be taken to guide the search in apparently profitable directions. Ultimately, a decision is taken to terminate the search, and the best solution visited is chosen.

Tabu search has been applied to many scheduling problems, see Glover and Laguna [44].

One of the most exciting modern tools for scheduling and timetabling is the genetic algorithm [Goldberg, 45; Davis, 46; Michalewicz, 47]. This mimics the process of evolution and survival of the fittest to move from a population of (usually) randomly generated starting solutions through many generations until hopefully some population contains a near-optimal solution. Much theoretical work has been undertaken on genetic algorithms, and some authors have tried rather slavishly to follow very closely the natural processes. Some of the most productive work in the scheduling field has, however, been achieved when researchers have followed broad evolutionary principles while inventing new reproductive procedures. Although all such processes are often referred to as genetic algorithms, the term *evolutionary algorithm* is perhaps a better description which does not tie processes too closely to actual genetics. Of the texts quoted above, Michalewicz perhaps best recognises this.

Scheduling work involving evolutionary algorithms includes Fang, Ross and Corne [48], Wren and Wren [49] and Kwan and Wren [50].

Constraint logic programming which controls heuristic search by skilful use of the constraints of a problem is another method now being used in many scheduling problems. Mention has already been made of the rostering work of Smith and Bennett [34]. Much of the current success of constraint programming is due to powerful commercial packages which help drive user-provided heuristics.

All the above methods have been applied to timetabling problems as evidenced in the current volume. They have also been used successfully in various job shop scheduling problems, while the travelling salesman problem has often been used to illustrate their potential. They have so far been less successful in the problem areas discussed in Sections 6 to 8 below, although their application to driver scheduling is being heavily researched by the author and colleagues. It is understood that constraint programming is being applied to rail locomotive scheduling in France, but there is reason to think that the successes reported may owe more to the embedded heuristics than to the constraint formulation.

Finally, in this review of processes, we refer to the ant system [Dorigo et al.,51]. This follows the processes used by ants which lay down trails of pheromone as they move between the nest and food sites. Initial moves are random, but ants moving along short paths return to the nest sooner, thereby strengthening the pheromone on the short trails sooner. Other ants are more likely to follow the strongest trails, so that these become ever stronger. The effects of pheromone gradually wear off, so that those trails which are used infrequently quickly lose their attractiveness. Dorigo et al. apply the system in the quoted paper to the travelling salesman problem, comparing it favourably to simulated annealing and tabu search. They also quote applications to other problems. This approach would appear to be very promising, and the present author has a research student experimenting with it on driver scheduling problems.

Some of the author's own work on scheduling is now reviewed, together with work on similar problems by others.

6. Rail Locomotive and Bus Scheduling

The simplest view of a locomotive scheduling problem is that a set of locomotives of a single class is to be allocated to a set of trains in such a way as to minimise the number of locomotives used, and subject to that minimum also to minimise the amount of light running (locomotives running on their own between arrival point of one train and the departure point of the next train allocated to the locomotive, or to and from a depot). The problem can be modelled in mathematical terms as a standard Assignment problem with a row and column for each train, and solved by the Hungarian algorithm. However, the number of trains in our first practical examples ranged up to 150, requiring cost matrices of 22.5K, well beyond our Ferranti Pegasus computer's capacity of 7K words for data and code together.

A heuristic [Wolfenden and Wren, 52] was therefore devised which had the merit of producing the optimal solution in all problems small enough to be checked against the Hungarian algorithm. In fact in the thirty years since the initial research, the heuristic has been applied to many thousand practical and often large rail and bus scheduling problems, and no better solution to real problems has ever been found. However, it is possible to construct artificial cases in which the constraint matrix of minimum journey times between points violates the triangular inequality and where the heuristic sticks in a local optimum. (Frequently bus companies have presented such constraint matrices to the system but have always agreed when the computer had checked them that they were illogical and have happily altered them.)

A strong lower bound to the required number of locomotives can easily be calculated based on the maximum number of simultaneously operating trains, where two trains are defined to be simultaneous if the locomotive used for the earlier finishing one cannot reach the starting point of any other train before the departure of the second. This bound mechanism was suggested by the author to Stern and Ceder who proved its validity [53]. A (possibly infeasible) solution is then developed using this bound as the number of locomotives; trains are taken in order of departure time and are assigned free locomotives rather arbitrarily; if no locomotive is free, then some locomotive is allocated infeasibly, and a penalty is assigned equal to the time needed to move the locomotive between the relevant points, minus the time available (which may be negative). The starting solution is then a linked list of activities (trains) for each locomotive where each link has either a cost equal to the time necessary to move between points or a penalty as above.

The heuristic then examines all pairs of links, exchanging their end points if this produces a reduction in total penalty, or a reduction in total cost for no increase in penalty. The process considers all possible pairs of links repetitively until two complete passes produce no improvement. An exchange is also made if no change in total cost or penalty results; this has the effect of kicking the current solution into a new pattern which can allow the heuristic to escape from a local optimum at the next pass.

Many experiments were carried out with different data sets and different starting solutions for each set, including some in which very large infeasibilities were present. All experiments, however bad the initial solution, converged for a given data set to the same total penalty and same total cost, apparently the global optimum. It may be noted, however, that without allowing exchanges which left penalty and cost unaffected, convergence would sometimes be to a local optimum.

Where a penalty remained after convergence, the original approach was to increase the number of locomotives by one and to try again. It was soon realised, however, that it was worth presenting the penalised links to the client, who would often find that it was possible to retime a train in order to remove the penalty. In the first practical exercise [52] implemented for British Railways (BR) in 1963 the number of locomotives was reduced from 15 to 12 (after two retimings) and the light running was reduced by 28%.

The first exercise had involved a set of freight trains ferrying traffic on the North London Line, principally between the docks and marshalling yards. These ran all round the clock (with similar work every weekday, and variations on Saturdays and Sundays), and the solution had to be adjusted to allow the locomotives to be returned from time to time to the depot for maintenance. This adjustment, which was carried out manually by the author, followed another heuristic which had been designed by him. The quality of the savings was preserved following the adjustment, although three additional locomotives were required to allow for maintenance, both in the "before" and "after" situations. Unfortunately, resources did not permit the coding (in Pegasus machine code) of the new heuristic which was quite complex.

It may be noted that the switching procedure is equivalent to pairwise swapping of the elements of a solution matrix for the assignment problem if the penalties are given precedence in the cost matrix. A major difficulty in having the process accepted by our peers in 1963 was the lack of a proof of optimality. The term "heuristic" was hardly known and the author was forced to use the term "trial and error" in presenting the process. By the time heuristics had become respectable, the method would have been classified as "eager but tedious" by Müller-Merbach [15].

The method was used by BR for several years, but was allowed to die when the Pegasus became obsolete, in favour of a system being developed in-house and based on linear programming. This was never fully implemented or published but was described by Wren [54] with the approval of its developers.

From 1970 the locomotive scheduling system became known as VAMPIRES and was adapted to bus operation, first in a simple single bus type, single depot situation (actually by separating out the work of different bus types from a single depot). The first trial was carried out for Yorkshire Traction in Barnsley where existing single decker and double decker work was treated separately and 23 buses were saved out of 107, partly by retiming as indicated by the system and partly by general improvements in efficiency (Wren [55]). Yorkshire Traction subsequently sponsored developments to allow more than one type of bus and more than one depot. Experiments for other companies followed, as outlined in [56] and [57]). The first regular user was however Greater Manchester Transport who installed the program in 1975 and used it as the basis of all their scheduling for several years. Buses are now scheduled in much smaller groups in response to competition, but the program is still used in Manchester on an occasional basis.

The consideration of more than one depot was relatively easy; an initial solution was built up as before, and the work of each bus was assigned to that depot providing the minimum cost links to the start of the first journey and from the end of the last journey. When a pair of links was then considered by the refining heuristic, the best depot after a prospective change was determined, and the total cost of links being exchanged and links to and from the depot was used to determine whether an exchange should be made. Special arrangements were made for buses that returned to the depot several times during the day. Although this process added to the computer time, the quality of the results was such as to make this acceptable.

The treatment of more than one vehicle type is too complex to consider fully here. The first tests were carried out on a problem involving nine basic types. A multi-stage process was devised wherein the best types of vehicle were considered when exchanges were examined. Several refining stages incorporating infeasibility of type as well as the earlier time infeasibilities were included. The solution of a classical Transportation Problem in which journeys were represented by columns and vehicle types by rows produced shadow costs which were used in subsequent phases to discourage the use of scarce bus types. The Transportation Problem was reformulated and solved again every time the heuristic converged, until either no bus had a journey for which its type was infeasible, or it could be concluded that there was no feasible solution, in which case either the number of buses was increased or management was invited to reconsider the constraints.

Although the results of this process were satisfactory where there were genuinely many types of vehicle, the increase in computer time was very great. Shortly after this development, there was a movement towards bus fleets consisting of a single vehicle type, and the multi-vehicle option was seldom used.

Although VAMPIRES was successfully used in many consultancy exercises, with considerable savings, its ability to explore all possible combinations of journey across a city or region was superfluous when routes were scheduled in relatively small groups, and operations were regular, so that the normal activity for a bus on reaching its destination was to turn and go back to its origin. A simpler system, called TASC [57], was developed in which dead journeys to other points were only considered if an arriving bus could not match a departure from the same point within a specified time.

A backtracking process was devised within TASC to try to ensure that where dead journeys were necessary, the most efficient set was used. TASC was installed for several bus operators in the early 1980's. It was easier to use than VAMPIRES, the data being specified as series of regular journeys on several routes, rather than as individual journeys, and was initially faster. However, users started to complain that sometimes the solution was not optimal, and the backtracking process was gradually made more complex to satisfy their complaints. Ultimately, a version of VAMPIRES was combined with TASC; a preliminary solution was found by TASC, and this was separated into blocks of continuous work with no long gaps or dead journeys; these blocks were then rescheduled by VAMPIRES as if they were individual journeys.

In 1983 the decision was made to convert the system for microcomputers, and after various trials were conducted it was determined that TASC had become so complex that VAMPIRES was now faster. Therefore the first microcomputer version contained a simple version of VAMPIRES which has since been made more complex. For example, a crude but fast method for dealing with more than one vehicle type was added at the request of a client; this incorporates a three-way swapping procedure. This micro version based on VAMPIRES is still in commercial use, although it has been suggested that it has become too complex, and that TASC should be resurrected, so that simple problems can be solved speedily!

7. Heuristics for Bus Driver Scheduling

That the bus driver scheduling problem could be formulated as a set covering or set partitioning problem was known from the early 1960's. However, no practical solution could be obtained from such a formulation until much later. Some unpublished work by Deutsch around 1963 sought to develop an exhaustive tree search, but was defeated by problems of any reasonable size. Throughout the period 1967-1978 the author and his associates devoted much attention to seeking a good heuristic method [58], culminating in the TRACS system [59]. Others (e.g., Elias, [60]) devised other heuristics throughout the 1960's.

Bus driver scheduling is the problem of designing legal shifts fitted to a day's bus schedule in such a way that all buses have a driver assigned at all times of day. Shifts have constraints on their duration, on the position of meal breaks and on other features which may be negotiated locally. Drivers can only be changed at *relief opportunities* when buses pass suitable *relief points.* Most shifts in the UK consist of two stretches of work separated by a meal break. While each stretch would ideally consist of work on a single bus, it is often necessary in order to fit all the bus work together or to cope with short peaked work that a stretch should contain more than one *spell,* the spells being on different buses and separated by small paid breaks. Although manually created shifts have been found with up to six spells, the author has never found a situation in which more than three spells was necessary in an efficient schedule. (Manual schedulers often have difficulty in fitting all the work together, and find it necessary to cut the last pieces into small spells and to distribute these among several shifts.)

Wren [61] and Wren and Rousseau [62] have surveyed computer methods for bus driver scheduling, many of which are now in regular use by bus companies.

When the author was first presented with this problem he hoped that a similar approach to that used in locomotive scheduling could be devised, i.e., that it would be possible to construct an infeasible schedule and to refine this until eventually a good feasible schedule was obtained. Unfortunately in removing the infeasibilities the schedule was forced to become progressively more fragmented. The fragmentation introduced wasteful gaps between spells of work, and the schedule therefore required too many shifts. Attempts to devise processes to move from this inefficient result to a significantly better one were unsuccessful.

Several attempts were made to obtain better starting solutions, and to devise suitable refining heuristics. These latter consisted of routines which exchanged work between shifts in several ways. While the refining routines were often (indeed usually) successful in improving existing manual schedules, it became evident that they were not capable of moving from a really bad schedule to a good one. In particular, although some of the routines were designed to reduce the number of shifts, these were seldom effectual, and when successful often produced a more fragmented schedule which could not be significantly improved. The conclusion that it was unlikely that heuristics could move from a poor solution to a good one was borne out by observation of manual schedulers; when a trainee constructed a schedule it would be checked by a senior colleague who would try to improve it; unless the first attempt was reasonably good, the senior scheduler would find it easier to start again than to adopt the poor schedule as a basis for adjustment.

Much time was therefore spent in improving the starting routine, and several approaches were designed and abandoned. Although it proved impossible to get schedulers to explain the processes they used to construct schedules, they generally accepted that the refining routines were similar to the processes they used to improve their first attempts. Examination of the problem structure enabled the team to identify certain critical features which would lead to inefficiency unless properly addressed, and by about 1974 they had designed a process which would produce reasonably good initial schedules for three sample bus operators. This process incorporated techniques gleaned from human schedulers, and would nowadays be described as an expert system, both knowledge-based and rule-based. The initial schedules were refined by the heuristics designed several years earlier, resulting in solutions which were at least as good as manually produced ones.

Over the next few years feasibility studies were carried out for some dozen bus operators. The constraints on schedule construction (shift length, etc.) vary greatly between companies, as do the "shapes" of underlying bus schedules. Initial experience showed that the heuristics behind the starting solution had to be adapted to deal with new situations, but that the adaptations could be designed and accommodated within a general method after about three person-months of research. We were therefore hopeful that we should ultimately develop a general heuristic for the initial process which would be applicable with minor, if any, modifications to new situations.

During this time, several organisations had implemented schedules produced by the system, and one had used the programs to experiment in conjunction with the unions with many different sets of possible constraints in order to design a new mutually acceptable set of rules. This last company entered negotiations to buy the system, which ultimately broke down when its in-house computer group objected to operating a system which could not guarantee an optimal solution, despite its consistently achieving better schedules than those produced manually.

The last three organisations for whom tests were undertaken upset the earlier confidence in the system, and no acceptable schedule was obtained within the budgeted three months. In Glasgow the length of the working day was greater than had been previously met, and a desired percentage of "early" shifts was not achieved. In Dublin the times of the peaks were much later than elsewhere. In London the bus schedule had always been carefully designed to accommodate a good driver schedule, and there was generally only one acceptable solution. We therefore initiated in 1978 a search for a different solution method. It may be remarked that Glasgow and London are now both using the resulting new system, while Dublin is using a similar one supplied by a Canadian rival.

8. Mathematical Programming Methods for Bus and Rail Driver Scheduling

Advances in computer power led us in 1978 to examine again the possibility of basing a driver scheduling system on integer linear programming. The author and a colleague [63], [64] and [65] describe a new system which was first installed for London Transport in 1984 and for many other organisations later. The basis of the method is the generation of many thousand potential shifts followed by a specialised integer linear programming (ILP) process to select a subset of shifts which form a good schedule, and a final set of refining heuristics which are similar to those used in the earlier research and which are able to restore shifts rejected by earlier processes.

Vital parts of this process are the use of heuristics to restrict the size of the generated set and to drive the branch and bound part of the ILP. These are too complex to describe here, and readers are referred to Smith and Wren [63] and to Smith [65] for further details. It is sufficient to say that the generation process is restricted by two sets of heuristics based on knowledge of the problem structure. The first removes relief opportunities which are unlikely to contribute to a good solution. Typically a quarter to a third of opportunities are removed, leading to a better than proportional reduction in generated shifts. The shift generation consists of an initialisation phase in which only potentially sensible shifts are chosen and a refining phase in which combinations of generated shifts are inspected and shifts which appear to be redundant are discarded. The ILP which follows takes advantage of the fact that the continuous relaxation rounded up if necessary almost always yields the correct total number of shifts, and uses all the relief opportunities required in the final solution. The original system is fully described in a thesis by Smith [66].

This system, originally known as IMPACS but in later versions as TRACS II, is now installed for about thirty transport operators, including bus, tram and light rail systems, and with extensions has been used in a feasibility study for London Underground, as well as in other studies for British Rail (see below). It can solve directly problems which are of a size to satisfy most British operators, who tend to schedule relatively small operating units separately. (Up to about 150 shifts are regularly solved satisfactorily.) However, some organisations do have larger problems, and a decomposition heuristic has been developed and is described by Wren and Smith [64]. This analyses the bus schedule and forms sub-problems consisting of buses which fit well together in a driver schedule; there is a carry-forward of inefficient shifts from one sub-problem to the next.

Other systems which use different mixtures of heuristics and mathematical programming are HASTUS [67] developed in the University of Montreal and HOT [68] implemented by Hamburger Hochbahn.

The TRACS II decomposition process has had very good results in practice, always beating manually produced schedules and those produced by rival systems in test cases. However, users are suspicious of decomposition, seeing its potential drawbacks and believing wrongly that a system which attempts to tackle the whole problem simultaneously must be better; in practice such systems appear to be further from the optimum than the results of decomposition.

Rail driver scheduling is considerably more complex than bus driver scheduling. Distances are greater, and drivers operate out of several different depots. Individual shifts may cover more separate trains than buses, so that potentially, many more shifts have to be generated for a given size of problem. Drivers frequently have to travel quite long distances as passengers in order to return to their home depot.

Recently, the TRACS II model has been adapted to provide a very good estimate of the numbers of drivers needed to cover any given train schedule under a wide range of labour conditions. This model [69, 70] has been used frequently since 1991 by the Operational Research Unit of British Rail to test different strategies. The model uses most of the features of TRACS II, but stops after the linear programming processes has reached a continuous solution, from which the estimate is developed by some further processes.

In 1994 the author and a colleague were awarded a contract to research and develop a rail driver scheduling system. This is being based on TRACS II, and at the time of writing successful schedules have been produced for several British railway systems. One of the operating companies has used the prototype system to generate full schedules under a number of different depot closure scenarios. Current research is directed at new heuristics for shift generation which will be suitable for a wider range of bus and rail problems and will overcome the problems caused by shifts covering more different trains; these are already leading to better results for rail problems than the previous method.

9. Modern Heuristics for Bus and Rail Driver Scheduling

Another approach worth considering for driver scheduling is the use of genetic algorithms. A feasibility study is described by Wren and Wren [49] in which an optimal solution was found to a small problem of fifteen shifts. The GA starts with an initial population of schedules formed by selecting shifts at random from the set generated by TRACS II, and replaces the ILP. A later study [71] failed to find good solutions to considerably larger problems, but research is continuing to refine the methodology.

Kwan and Wren [50] are currently experimenting with another approach in which a GA is used to assist in determining the relief opportunities to be rejected in TRACS II or in any other driver scheduling system. A number of sample sets of chosen relief opportunities provide an initial population, each member of which is costed by applying the estimator described above. (A good set should provide a low estimate of shift numbers.) Mating of population members proceeds in a suitable way until hopefully a set of relief opportunities is obtained which yields an estimate equivalent to the optimum solution. It should be possible to generate from these a good set of shifts to provide the basis either for the ILP of TRACS II or for a GA approach as outlined above. As the set of relief opportunities obtained in this way should be much smaller than that currently used in TRACS II it should be possible to solve much larger problems, and initial investigations have proved encouraging.

Other methods being applied to bus and train driver scheduling in new research by the author and colleagues are tabu search (possibly to resolve certain combinatorial sub-problems affecting the structure of shifts at certain times of day and therefore to cut down the number of potential shifts which need to be considered), constraint programming (for the same sub-problems), and a new approach based on the ant system [51].

10. Conclusions

A large number of methods has been applied to a range of timetabling and scheduling problems, with varied success. A common strand in these developments runs through simple heuristics in the 1960's and 1970's, with some mathematical models formulated but not applied, integer linear programming driven by heuristics through the 1980's and a range of modern metaheuristics in the 1990's.

Many problems are now being satisfactorily solved, but in some areas there is still a gap between theory and practice (relatively few large universities, for example, are committed to existing timetabling systems, although most are actively seeking good systems).

Although similar methods have been tentatively applied to all the types of problem discussed in this paper, their degree of success has varied considerably across the problem types. Frequently researchers have attempted to carry over approaches from one type to another, with little, if any, reported success. However, the points of similarity between timetabling and staff rostering are such that there may be scope for cross-fertilisation. Driver scheduling (i.e. shift construction) and the various routeing problems appear to be very different structurally from timetabling.

While useful methods are regularly employed for some problem types, for example in bus driver scheduling where systems are being used in many hundred or thousand organisations, even then there is much scope for improvement, and developments in the methodologies itemised in Section 5 should lead to greatly improved systems over the next few years.

References

1. A. Wren (ed.), Computer scheduling of public transport. North-Holland (1981).

2. J-M. Rousseau (ed.), Computer scheduling of public transport - 2. North-Holland (1985).

3. J.R. Daduna and A. Wren (eds.), Computer-aided transit scheduling. Springer-Verlag (1988).

4. M. Desrochers and J-M. Rousseau (eds.), Computer-aided transit scheduling, 2. Springer-Verlag (1992).

5. J.R. Daduna, I. Branco and J.M.P. Paixao (eds.), Computer-aided transit scheduling, 3. Springer-Verlag (1995).

6. S. French, Sequencing and scheduling. Ellis Horwood (1982).

7. M. Carey, A model and strategy for train pathing with choice of lines, platforms and routes. *Transp. Research*, 28B, 333-353 (1994).

8. K. Wolfenden and A. Wren, Locomotive scheduling by computer. *In* Proceedings of the British Joint Computer Conference, *IEE Conference publication* 19, 31-37 (1966).

9. A. Wren, Bus scheduling, an interactive computer method. *Transportation Planning and Technology*, 1, 115-122 (1972).

10. P.D. Manington and A. Wren, Experiences with a bus scheduling algorithm which saves vehicles. *Pre-prints of* International Workshop on Urban Passenger Vehicle and Crew Scheduling, Chicago (25 pp.) (1975).

11. B.M. Smith and A. Wren, VAMPIRES and TASC: two successfully applied bus scheduling programs. *In* A. Wren (ed.) *Computer scheduling of public transport.* North-Holland, Amsterdam, 97-124 (1981).

12. S.M. Johnson, Optimal two- and three-stage production schedules with set-up times included. Nav. Res. Logist. Q., 1, 61-68 (1954).

13. J.M. Moore, An n-job, one machine sequencing algorithm for minimising the number of late jobs. Mgmt. Sci., 15, 102-109 (1968).

14. E.L. Lawler, Optimal sequencing of a single machine subject to precedence constraints. Mgmt. Sci., 19, 544-546 (1973).

15. H. Müller-Merbach, Heuristic methods: structures, applications, omputational experience. *In* R. Cottle & J. Krarup (eds.) *Optimisation Methods for Resource Allocation*, English Universities Press, 401-416 (1974).

16. S. Lin, Computer solution of the travelling salesman problem. *Bell System Technical Journal*, 44, 2245-2269 (1965).

17. G.B. Dantzig and J.H. Ramser, The truck dispatching problem. *Man.Sci.* 6, 80-91 (1959).

18. G. Clarke and J.W. Wright, Scheduling of vehicles from a central depot to a number of delivery points. *Opns.Res.* 12, 568-581 (1964).

19. T.J. Gaskell, Bases for vehicle fleet scheduling, *Opl.Res.Q.*, 18, 281-295 (1967).

20. N. Christofides and S. Eilon, An algorithm for the vehicle-dispatching problem. *Opl.Res.Q.*, 20, pp.309-318 (1969).

21. A. Wren, Applications of computers to transport scheduling in the United Kingdom, chapter 10, pp.70-75. West Virginia University Engineering Experiment Station Bulletin 91 (1969).

22. A. Wren and A. Holliday, Computer scheduling of vehicles from one or more depots to a number of delivery points. *Opl.Res.Q.* 23, 333-344 (1972).

23. B.C. Gillett and L.R. Miller, A heuristic algorithm for the vehicle dispatch problem. *Ops.Res.*, 22(2), 340-349 (1974).

24. B.A. Foster and D.M. Ryan, An integer programming approach to the vehicle scheduling problem. *Opl.Res.Q.* 27, 367-384 (1976).

25. G. Laporte, The vehicle routing problem: an overview of exact and approximate algorithms. Université de Montréal, Centre de Recherche sur les Transports, Publication 745 (1991).

26. R.S.K. Kwan, Co-ordination of joint headways. *In* J.R. Daduna and A. Wren (eds.) *Computer-Aided Transit Scheduling*. Springer-Verlag, Berlin, 304-314 (1988).

27. W. Keudel, Computer-aided line design (DIANA) and minimisation of transfer times in networks. *In* J.R. Daduna and A. Wren (eds.) *Computer-Aided Transit Scheduling*. Springer-Verlag, Berlin, 315-326 (1988).

28. R. Günther, Untersuchung planerischer und betrieblicher Maßnahmen zur Verbesserung der Anschlussicherung in städtischen Busnetzen. Schriftenreihe des Instituts für Verkejrsplanung und Verkehrswegebau der Technischen Universität Berlin (1985).

29. A.J. Cole, The preparation of examination timetables using a small store computer. *Computer Journal*, 7, 117-121 (1964).

30. D.C. Wood, A system for computing university examination timetables. *Computer Journal*, 11, 41-47 (1968).

31. A.W. Parker, M.E. Parker and L.G. Proll, Constructing timetables for parent-teacher interviews - a practical scheduling problem. Preprints of Combinatorial Optimisation 81 (CO81) 122-137 (1981).

32. B.T. Bennett and R.B. Potts, Rotating roster for a transit system. *Transpn.Sci.* 2, 14-34 (1968).

33. W. Townsend, Bus crew rostering by computer. University of Leeds MSc thesis (1985).

34. B.M. Smith and S. Bennett, Combining constraint satisfaction and local improvement algorithms to construct anaesthetists' rotas. Proc. Conference on Artificial Intelligence Applications (CAIA 92), 106-112 (1992).

35. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller, Equation of state calculation by fast computing machines. *J. of Chem. Phys.*, 21, 1087-1091 (1953).

36. S. Kirkpatrick, C.D. Gellatt and M.P. Vecchi, Optimization by simulated annealing. *Science*, 220, pp.671-680 (1983).

37. S. Kirkpatrick and G.B. Sorkin, Simulated annealing. *In* M. Arbib (ed.), *Handbook of brain theory and neural networks.* MIT Press (1995).

38. N.E. Collins, R.W Eglese and B.L. Golden, Simulated annealing - an annotated bibliography. *AJMMS*, 8, 209-307 (1988).

39. J. Thompson and K.A. Dowsland, Variants of simulated annealing for the examination timetabling problem. *Annals of Operations Research* (1995).

40. P.M. Ross and D. Corne, Comparing genetic algorithms, stochastic hillclimbing and simulated annealing. *In* T.C.Fogarty (ed), *Evolutionary computing*, Springer-Verlag, 94-102 (1995).

41. I.H. Osman, Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research,* 41 (1993).

42. F. Glover, Tabu search - Part 1. *ORSA J. Computing*, 1, 190-206 (1989).

43. F. Glover, Tabu search - Part 2. *ORSA J. Computing*, 2, 4-32 (1990).

44. F. Glover and M. Laguna, Tabu search. *In* C.R. Reeves (ed.) *Modern heuristic techniques for combinatorial problems.* Blackwell Scientific Publications, 70-150 (1993).

45. D.E. Goldberg, Genetic algorithms in search, optimisation and machine learning. Addison-Wesley (1989).

46. L. Davis, Handbook of genetic algorithms. Van Nostrand Reinhold (1991).

47. Z. Michalewicz, Genetic algorithms + data structures = evolution programs, second, extended edition. Springer-Verlag (1994).

48. H-L Fang, P.M. Ross and D.Corne, A promising genetic algorithm approach to job-shop scheduling, rescheduling and open-shop scheduling problems. *In* S. Forrest (ed.) *Proc. 5th International Conference on Genetic Algorithms.* Morgan Kaufmann, 375-382 (1993).

49. A. Wren and D.O. Wren, A genetic algorithm for public transport driver scheduling. *Computers Ops Res.* 22, 101-110 (1995).

50. R.S.K. Kwan and A. Wren, Hybrid algorithms for bus driver scheduling. *To appear in* L. Bianco and P. Toth (eds.) *Advanced methods in transportation analysis*, Springer-Verlag (1996).

51. M. Dorigo, V. Maniezo and A. Colorni, The ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B*, 26, pp.1-13 (1996).

52. K. Wolfenden and A. Wren, Locomotive scheduling by computer. Proc. British Joint Computer Conference. *IEE Conference Publication 19*, 31-37 (1966).

53. H.I. Stern and A. Ceder, An improved lower bound to the minimum fleet size problem. *Transpn.Sci.* 17, 471-477 (1983).

54. A. Wren, *Computers in Transport Planning and Operation.* Ian Allen, London, 103-106 (1971).

55. A. Wren, Bus scheduling, an interactive computer method. *Transportation Planning and Technology*, 1, 115-122 (1972).

56. P.D. Manington and A. Wren, Experiences with a bus schedulionq algorithm which saves vehicles. *Preprints of* International Workshop on Urban Passenger Vehicle and Crew Scheduling, Chicago (25 pp) (1975).

57. B.M. Smith and A. Wren, VAMPIRES and TASC: two successfully applied bus scheduling programs. *In* A. Wren (ed.) *Computer scheduling of public transport.* North-Holland, Amsterdam, 97-124 (1981).

58. B. Manington and A. Wren, A general computer method for bus crew scheduling. *Pre-prints of* International Workshop on Urban Passenger Vehicle and Crew Scheduling, Chicago (49 pp) (1975).

59. M.E. Parker and B.M. Smith, Two approaches to computer crew scheduling. *In* A. Wren (ed.) *Computer Scheduling of Public Transport.* North-Holland, Amsterdam, 193-222 (1981).

60. S.E.G. Elias, The use of digital computers in the economic scheduling for both man and machine in public transportation. *Kansas State University Bulletin, Special Report* 49 (1964).

61. A. Wren, General review of the use of computers in scheduling buses and their crews. *In* A. Wren (ed.) *Computer Scheduling of Public Transport.* North-Holland, Amsterdam, 3-17 (1981).

62. A. Wren and J-M. Rousseau, Bus driver scheduling - an overview. *In* J.R. Daduna, I. Branco and J.M.P. Paixao (eds.) *Computer aided transit scheduling - 3*, Springer-Verlag, Berlin, 173-187 (1995).

63. B.M. Smith and A. Wren, A bus crew scheduling system using a set covering formulation. *Transpn.Res.* 22A, 97-108 (1988).

64. A. Wren and B.M. Smith, Experiences with a crew scheduling system based on set covering. *In* J.R. Daduna and A. Wren (eds.) *Computer-aided transit scheduling.* Springer-Verlag, Berlin, 104-118 (1988).

65. B.M. Smith, IMPACS - a bus crew scheduling system using linear programming. *Math Prog.* 42, 181-187 (1988).

66. B.M. Smith, Bus crew scheduling using mathematical programming. University of Leeds PhD thesis (1986).

67. J-M. Rousseau and J-Y Blais, HASTUS: an interactive system for buses and crew scheduling. *In* J-M Rousseau (ed.) *Computer Scheduling of Public Transport - 2.* North-Holland, Amsterdam, 45-60 (1985).

68. J.R. Daduna and M. Mojsilovic, Computer-aided vehicle and duty scheduling using the HOT programme system. *In* J.R. Daduna and A. Wren (eds.) *Computer-Aided Transit Scheduling.* Springer-Verlag, Berlin,133-146 (1988).

69. A. Wren, R.S.K. Kwan and M.E. Parker, Scheduling of rail driver duties. In T.K.S. Murthy et al. (eds.) *Computers in railways IV - Volume 2, Railway Operations*, 81-89 (1994).

70. M.E. Parker, A. Wren and R.S.K. Kwan, Modelling the scheduling of train drivers. *In* J.R. Daduna, I. Branco and J.M.P. Paixao (eds.) *Computer aided transit scheduling - 3*, Springer-Verlag, Berlin, 359-370 (1995).

71. R.P. Clement and A. Wren, Greedy genetic algorithms, optimizing mutations and bus driver scheduling. *In* J.R. Daduna, I. Branco and J.M.P. Paixao (eds.) *Computer aided transit scheduling - 3*, Springer-Verlag, 213-235 (1995).