UNIVERSITY OF LEEDS

This is a repository copy of *Driver scheduling by integer linear programming: the TRACS II approach*.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/92942/

Version: Accepted Version

Proceedings Paper:
Fores, S and Proll, L (1998) Driver scheduling by integer linear programming: the TRACS II approach. In: Borne, P, Ksouri, M and El Kamel, A, (eds.) Proceedings CESA'98 Computational Engineering in Systems Applications. Symposium on Industrial and Manufacturing Systems, 01-04 Apr 1998, Nabeul Hammamet, Tunisia. CESA , pp. 213-218.

This is an author produced version of a paper published in Proceedings CESA'98 Computational Engineering in Systems Applications.

Reuse
Unless indicated otherwise, fulltext items are protected by copyright with all rights reserved. The copyright exception in section 29 of the Copyright, Designs and Patents Act 1988 allows the making of a single copy solely for the purpose of non-commercial research or private study within the limits of fair dealing. The publisher or other rights-holder may allow further reproduction and re-use of this version - refer to the White Rose Research Online record for this item. Where records identify the publisher as the copyright holder, users can verify any specific terms of use on the publisher's website.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

Driver Scheduling by Integer Linear Programming - The TRACS II Approach

Sarah Fores and Les Proll
Scheduling and Constraint Management Group
School of Computer Studies
University of Leeds
Leeds LS2 9JT
UK

1 INTRODUCTION

Vehicle and driver scheduling problems have been tackled using computers since the early 1960's. Many different techniques developed with improvements in technology but mathematical programming approaches are now the most dominant and successful. Several complete driver scheduling packages are commercially available and some details of these and other investigations have been published as a result of seven international workshops (see, e.g.,[1, 2]). One such package which uses a mathematical programming approach is TRACS II which was originally developed from the IMPACS system [3] and is now used within both the UK bus and train industries.

This paper concentrates on the mathematical programming component of the TRACS II system, describing the algorithm and recent developments. Results on a selection of real-world bus and train problems are reported.

2 THE DRIVER SCHEDULING PROBLEM

Although some systems attempt to schedule vehicles and drivers simultaneously, or even to schedule drivers first, it is preferable first to build a vehicle schedule to cover a set of predetermined journeys. This gives the user the flexibility of making any changes to the vehicle schedule first, although it loses the opportunity of joint optimality where moving a vehicle a few minutes earlier or later may result in saving a driver because of the way the work can be combined. The vehicle schedule is represented graphically by a series of lines each of which depict the movements of a vehicle during the day, and at various stages the vehicle will pass a location which is convenient for driver changeovers. These location/time pairs are known as *relief opportunities*, and an indivisible period between any two relief opportunities is known as a *a piece of work*. The work of a driver in a day is known as *a shift*, which usually consists of two to four spells of work each of which cover several consecutive pieces of work on the same vehicle. The formation of shifts is governed by a set of labour agreement rules to ensure that there is adequate provision for mealbreaks and acceptable working hours etc. The driver scheduling problem is then to assign drivers in such a way that all the vehicle work is covered and some measure of the number of drivers and the costs of shifts is minimised.

It is possible to formulate the driver scheduling problem as a set covering or set partitioning problem which ensures that all of the vehicle work is covered. A suitable objective function can then be devised which ensures shift and cost minimisation over a set of previously generated valid shifts. The difficulty with this approach is that the number of valid shifts is too large for this approach to be able to guarantee an optimal schedule in most cases. Thus mathematical programming is frequently combined with heuristic approaches to provide a viable solution method.

3 THE TRACS II SYSTEM

TRACS II is the driver scheduling system developed by the SACM group at the University of Leeds. Many of the algorithms have been improved since its origins as the IMPACS system and more recently it has been adapted to incorporate features required in order to schedule train crews [4]. There are essentially three stages to the solution method:

- **Stage 1** *Generate a set of shifts which are valid according to labour agreement rules.*
  It should be noted that in a practical problem there are generally many million potential shifts. The shift generation process only produces a large subset of shifts, chosen heuristically in such a way that the most likely shifts are formed and that a good choice of shifts is available for each piece of work. These heuristics have proved effective in a wide range of practical applications
- **Stage 2** *If necessary, reduce the size of the generated shift set.*
  Where a standard ILP approach is used it optimises over the whole shift set, and this is limited in terms of data storage

and is time-consuming, even with improvements in technology and algorithms. Heuristic methods have been developed which attempt to reduce the size of the set, while retaining the best possible subset of shifts. It should be noted that one approach now used in TRACS II is column generation which can handle much larger shift sets.

- **Stage 3** *Select from the shift set a subset which covers the vehicle work.*
  The problem is one of minimising the overall schedule cost, which includes a wage cost and sometimes a subjective cost reflecting penalties for shifts containing undesirable features. Since the size of shift set originally generated is limited, it may not be possible to use a set partitioning approach which would imply that each piece of work should be covered by exactly one driver. A set covering model is used which guarantees that each piece of work is covered by at least one driver and the details of this method are discussed in the following sections.

4 THE TRACS II DEFAULT MODEL

Given a problem with M pieces of work in the vehicle schedule, and a previously generated set of N shifts, we can define :

$$\text{For } j = 1, .., N$$

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

(4.1)

4.1 The Objective Function

Several different overall objectives are available to users. Since the introduction of each driver incurs a large cost it is normally the case that the reduction of drivers has a higher priority than the reduction of costs, but users who wish to retain and utilise fully their current workforce can choose only to minimise cost. Generally users wish to minimise the number of drivers and within that target then minimise any further costs such as penalties for shifts which contain undesirable features.

The objectives can be combined using a Sherali weighted objective function [5] of the form :

$$\text{Minimise} \quad \sum_{j=1}^{N} D_j x_j$$

where

D_j	$= W + C_j$ for $j = 1, .., N$
C_j	combines wage and penalty costs for shift$j$
W_1	$= 1 +$ sum of X largest C_j values
X	is the number of shifts in the initial solution

(4.2)

The Sherali weight W must be sufficiently large to ensure that the reduction of shifts is given priority over the reduction of other costs in order to guarantee the minimum number of shifts in the solution. The choice of W above ensures this since the number of shifts in the initial solution is an upper bound on the total number of shifts in the schedule.

4.2 Constraints

As mentioned earlier the model is one of set covering, because the number of shifts generated is limited to those which are deemed to be 'efficient' and many shorter shifts are discarded. The following constraint ensures that every piece of work is covered by at least one driver:

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

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

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

(4.3)

Overcover occurs where more than one driver is allocated to a piece of work. In practice this can be allowed by having a set-covering model, but is minimised because it would incorporate more than one wage cost. Any remaining overcover then occurs around the middle of the day and can be removed by editing shifts. Thus workpiece constraints corresponding to first pieces of work on early departing vehicles and to last pieces of work on late arriving vehicles are defined as equations. The inclusion of such constraints is, in fact, a requirement for the validity of the constraint branching strategy used within the branch and bound algorithm. The algorithm incorporated into the mathematical programming component automatically identifies the workpieces which shall be formulated as equality constraints, ensuring that the sets of shifts covering each of these workpieces are mutually exclusive.

Hence, constraint **(4.3)** can now be split into the following:

$$\begin{aligned} \textstyle\sum_{j=1}^{N} A_{ij}x_j &= 1 \text{ for } i = 1,..,L \\ \textstyle\sum_{j=1}^{N} A_{ij}x_j &\geq 1 \text{ for } i = L+1,..,M \end{aligned} \qquad \textbf{(4.4)}$$

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

4.2.1 Side Constraints: Shifts can be categorised by an initially specified type and the user can impose constraints to ensure that the final schedule does not contain too many or too few shifts of any particular type. Constraints of this form can also be used to limit the total number of shifts in the final schedule. The constraints can be imposed either initially or when reoptimising an existing solution.

The constraints can be expressed as follows:

$$\textstyle\sum_{j=1}^{N} \delta_{kj}x_j \leq U_k \quad , \quad \sum_{j=1}^{N} \delta_{kj}x_j \geq L_k \qquad \textbf{(4.5)}$$

where U_k is an upper limit on the number of shifts of type k, and L_k is a lower limit on the number of shifts of type k.

The δ_{kj} are defined as :

$$\delta_{kj} = \begin{cases} 1 & \text{if } x_j \text{ is a shift of type k} \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{(4.6)}$$

In summary, the model can be defined as :

$$\begin{aligned} \text{Minimise} \quad & \textstyle\sum_{j=1}^{N} D_j x_j \\ \text{Subject to} \quad & \textstyle\sum_{j=1}^{N} A_{ij}x_j = 1 \text{ for } i = 1,..,L \\ & \textstyle\sum_{j=1}^{N} A_{ij}x_j \geq 1 \text{ for } i = L+1,..,M \\ & \text{Plus any side constraints} \\ & x_j = 0 \text{ or } 1, \text{ for } j = 1,..,N. \end{aligned} \qquad \textbf{(4.7)}$$

5 METHOD OF SOLUTION (SLPZIP)

It is now possible to solve the model **(4.7)** in many different ways according to some user-defined parameters. Some choice is not apparent to the user because its complex nature was intended for use by the developers in solving problems which do not behave as expected when run through the default settings. Users do have control over where the solution method starts and ends, e.g., picking up previous solutions or only solving to the LP relaxation. Here we describe the complete solution

method. Some of the other techniques governed by user-parameters will be described in section 5.6.

There are in fact two default solution techniques depending on the number of shifts available to the mathematical programming component. A dual steepest edge method is used for sets containing up to 30,000 shifts and column generation best handles larger shift sets. This section details the two solution techniques and some of the processes contained within them.

5.1 Column Generation

Driver scheduling problems are a class of combinatorial optimisation problems which contain many variables. Column generation has been successfully applied to problems which would otherwise have to be reduced or decomposed before being solved. In order for a conventional mathematical programming approach to be used to solve the driver scheduling problem, heuristics are necessary to cut down the problem size. This restricts optimality to only those shifts remaining. The heuristics have been developed so that apparently inefficient shifts are removed, and solutions have proved better than manual solutions. However it may be the case that some inefficient shifts would link well with other shifts to produce the optimal solution. In the context of driver scheduling a column generation method implicitly considers many more valid shifts than standard linear programming approaches whilst retaining a much smaller working subset.

In order to guarantee an optimal solution with a column generation process it is necessary to ensure that either all possible valid shifts have been generated previously or that the technique of 'generating' further shifts is capable of considering all possible valid shifts. The Crew-Opt [6] module contained within the HASTUS scheduling system [7] generates new shifts using a constrained shortest path algorithm within its mathematical programming component. Since TRACS II already has a good shift generation process, and the shift costs include penalty costs which are subjective and complicated to model using the standard shortest path generation technique, a method has been implemented which considers a previously generated shift superset and allows many more shifts to be available to the mathematical programming solver. Although the continuous solution is still not guaranteed to be optimal as all possible shifts have not been considered, better solutions lead to a branch and bound search for schedules with fewer shifts. A subset of the shifts is available at the outset. The Revised Simplex Method is used to find the solution which is optimal over the subset. One then searches through shifts not previously considered, by means of an enumeration method, and adds some or all shifts with favourable reduced costs as non-basic variables. The new subset is then reoptimised. Given any LP relaxation which is optimal over its available subset the overall optimal solution is attained when no more shifts which would improve the objective can be added to the subset.

Fores [10] details the column generation implementation strategies. The solution method is outlined as follows:

- **Step 0: Generate a shift superset**

  This uses the TRACS II technique of generating shifts, possibly allowing more shifts to be produced by relaxing some of the conditions set by earlier implementations to restrict the shift generation. No further heuristic reduction is then necessary.

- **Step 1: Create an initial solution and form an initial shift subset**

  The method of finding an initial solution is that of generating one from the shift superset, using the method described in section 5.3. As shifts are being considered in forming the initial solution a shift subset is also being selected. This subset needs to be sufficiently large and varied to reduce the number of shift additions required.

- **Step 2: Solve the LP relaxation over the current shift subset**

  Use a primal steepest edge [8] variant of the Revised Simplex Method to solve the LP over the current shift subset.

- **Step 3: Add a set of shifts to the current subset which will improve the solution**

  Simplex multipliers produced at the end of Step 2 are used to calculate the reduced costs of shifts currently not selected from the superset.

  The reduced cost of a shift k is defined to be :

$$\text{MIN } (D_k - \sum_{i=1}^{M} \pi_i a_{ik}). \tag{4.8}$$

where :
M is the number of constraints
D_k is the cost of shift k
π_i is the simplex multiplier for constraint i
a_{ik} is the coefficient of shift k in constraint i
Since the shift costs include the large Sherali weight they do not vary as significantly as the simplex multipliers. The simplex multipliers are therefore considered in decreasing order and shifts covering their corresponding pieces of work are added to the subset if they have a favourable reduced cost and if they are allowed to be added according to several parameters which control the shift addition.

- **Step 4: If favourable shifts can be found go to Step 2**

- **Step 5: Determine the target number of shifts**

  If the total number of shifts in the LP solution is of the form $I.f(0 < f < 1)$, the side constraint

$$\sum_{j=1}^{N} x_j \quad \geq I + 1 \tag{4.9}$$

  is added and the model re-solved using a dual simplex algorithm.

- **Step 6: Reduce the problem size in terms of variables and constraints.** (See section 5.4)

- **Step 7: Find an integer solution using branch and bound.** (See section 5.5)

The shift superset is stored on a random access file and thus is essentially unlimited in size. The current subset is held in memory and consequently its maximum size subset is limited (currently to 30,000 shifts) which may result in a premature exit from Step 4. A table of pointers to the shifts in the superset which cover each piece of work enables Step 3 to be performed efficiently.

5.2 A Dual Approach

It has long been known that LPs of the form **(4.7)** are inherently degenerate and that primal simplex approaches to their solution spend many iterations making no reduction in the value of the objective function. The effect of degeneracy can be lessened by a dual simplex approach. In order to use a dual approach, the initial solution (see section 5.3) has to be transformed into a basic dual feasible solution, i.e. one in which no reduced costs are favourable. We then find the optimal solution to the LP relaxation using a dual steepest edge algorithm [8]. Willers' experiments [9] showed an average reduction in solution time of 51% over the primal steepest edge approach.

Problems of up to 30,000 shifts can be handled comfortably in the memory available in most client machines. Dual steepest edge requires all shifts to be held in memory. Experiments have shown that for problems with smaller shift sets, e.g. under 20,000 which was the limit under earlier versions of TRACS II, dual steepest edge generally solves faster than column generation. The shift size for which the column generation method becomes generally faster is not yet known (although it can be proved that it is quicker on larger sets) but the physical limit of 30,000 shifts seems a plausible point at which to change to column generation.

5.3 Initial Solution

Finding an initial solution to **(4.7)** is necessary to determine the Sherali weights and to reduce the time taken to solve the LP relaxation. It is constructed by a heuristic which sequentially looks at the currently uncovered piece having the least shifts available to cover it. From these, we select a shift by:

$$\text{Max} \quad \sum_{i=1}^{M} \Delta_{ij} L_i \tag{5.1}$$

where

$$\Delta_{ij} = \begin{cases} 1 \text{ if shift j covers the currently} \\ \text{uncovered piece of work i} \\ 0 \text{ otherwise,} \end{cases}$$

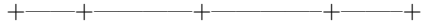
L_i = duration of workpiece i.

i.e. the shift covering the most currently uncovered work is selected, thus attempting to cover all work with a small number of shifts. To provide an advanced start for the solution of the LP the heuristic solution has to be transformed into a basic solution to **(4.7)**. This solution may not be a feasible one, particularly in the presence of side constraints. If this is the case, a Phase 1 procedure is invoked.

5.4 Reduce

The REDUCE procedure exploits the idea that the LP optimum gives a good indication of how the vehicle work would be covered in a schedule in order to reduce the size of the problems to be solved in the branch and bound phase. The principle is to restrict the choice of relief opportunities to those associated with shifts selected by the LP optimum.

The majority of the constraints in **(4.7)** are associated with pieces of work, each of which starts and ends at a relief opportunity. The vehicle block



where + denotes a relief opportunity and - denotes a vehicle movement, would generate four workpiece constraints. The first and last relief opportunity on any block must be used by at least one of the shifts in the LP solution, since it covers all work pieces. Thus any unused relief opportunity separates two pieces of work. Suppose the third relief opportunity in the above running board is unused and is eliminated from the problem. The running board would change to

and would generate only three workpiece constraints. Thus eliminating an unused relief opportunity allows one of the two workpiece constraints surrounding it to be removed. In addition any shift which does not use one of the selected relief opportunities can be deleted. Willers [9] provides an algorithm for constructing an optimal LP basis to the reduced problem from the optimal LP basis to the original problem.

It is possible that the constraint limiting the search to find a schedule with an exact number of shifts may not be satisfied with the limited set of relief opportunities. However, extensive practical experience suggests that this happens rarely and is outweighed by the reduction in the solution time for the branch and bound phase resulting from reductions in the number of workpiece constraints of 13-50% and in the number of shifts of 50-90%.

5.5 Branch and Bound

The branch and bound method has been developed with an emphasis on finding a good integer solution quickly and uses a specialised hierarchy of branching strategies developed by Smith and Wren [3]. The schedule is found by developing a branch and bound tree, in which the lower bound on the objective cost is given by the optimal continuous solution. Once an integer solution has been found the nodes of the tree are fathomed if their cost is greater than or equal to a scheduler specified percentage of the current best integer cost. In most cases the first integer solution found fathoms all remaining active nodes and hence the branch and bound process normally terminates with a possibly non-optimal integer solution. The tree is held in memory and allows a maximum of 500 nodes. In the event that the branch and bound terminates without finding a schedule, an internal looping mechanism allows the target number of shifts to be increased and the branch and bound process restarted. The default optimisation algorithm within the branch and bound process is dual steepest edge.

5.6 User Controllability

SLPZIP has been designed to be flexible and to allow a variety of different solution strategies. The impetus for this is that, particularly when TRACS II is being introduced into a new organisation, several runs are frequently necessary to obtain an acceptable schedule. This is, in part, due to the fact that some of the constraints governing the validity of shifts are 'soft' in nature and, in part, due to the heuristics used to generate the shifts.

The operation of SLPZIP is driven by two parameter files, one mandatory and one optional. The mandatory file allows the user to control whether the process starts from scratch or from a previous solution, the fathoming tolerance to be used in branch and bound, the extent of internal looping, whether or not to use the REDUCE procedure and the costs to be used. The optional file is intended primarily for developers rather than end users and controls the algorithmic path through SLPZIP, including the initial solution strategy, the variants of the simplex method used in both the LP relaxation and the branch and bound phase and the parameters controlling shift additions in column generation. In the absence of this file, default algorithmic settings are employed.

6 COMPUTATIONAL RESULTS

One of the objectives in developing the SLPZIP system was to allow several solution techniques to be available to the user. The viability of each method has been tested independently and it is infeasible to compare the quality and solution speed of each of the numerous combinations. Also, developments have taken place in parallel on different platforms and have been compared with versions in various stages of improvement. The principal interest is in the performance of the column generation approach since, in principle, it can provide better schedules. Table 1 compares the results obtained using the dual approach with those obtained using the column generation approach. The shift set submitted to the column generation approach is that obtained from the BUILD component of TRACS II [4]; that tackled by the dual approach is a subset of this set, limited to at most 30,000 shifts, obtained by passing the set through the SIEVE component [4]. Timings are reported in minutes for a Silicon Graphics Iris Indigo Workstation with 33MHz R3000 MIPS Processor, except for the final two results which were achieved on a 200MHz PC.

Data Set	Standard (S)			Column Generation (L)		
	Shifts	run time		Shifts	run time	
		LP	total		LP	total
AUC	87	13	36	87	21	91
CTJ	88	10	16	87	9	19
CTR	–	–	–	88	18	54
GMB	34	0.6	0.9	33	0.7	0.9
RI2	45	0.9	2.2	44	1.0	1.2
STK	61	5.2	16	59	21	24
SYD	56	5.4	5.8	56	15	18
UBAG	112	15	33	111	40	42
L99	247	18	29	245	48	55

Table 1: Comparison of Solutions and Timings

It can be seen from Table 1 that the number of shifts required was reduced in 7 of the 9 sets. This includes the one problem where no solution could previously be found. It is difficult to compare timings through a branch and bound tree because the route through it will be different. However, it is useful to note that where there is an increase in the overall execution times, the times themselves would still be acceptable to users.

Table 2 shows the maximum size reached by the working subset in four recent scheduling exercises, together with the number of shift addition iterations (section 5.3 Step 3) performed by the column generation algorithm and the number of shifts remaining after the REDUCE process. The fact that the subset size is only of the order of 20% of the superset size suggests that the rules used for shift addition are intelligent and that it is feasible to tackle even larger superset sizes. It may also be noted that the maximum allowed subset size of 30,000 is not reached

When the same shift set is used, Fores' results [10] show an average reduction in execution time of 41% for column generation over primal steepest edge compared with the 51% reduction using dual steepest edge. Thus column generation is almost competitive with the dual approach on problems of up to 30,000 shifts.

7 CONCLUSION

The TRACS II system is in extensive use in the UK bus and train industries and consistently provides better schedules than those in use. There are still limitations to the system in terms of the size of the vehicle schedule which it is sensible to process,

Data Set	Shifts in Superset	Maximum Subset Size	No. Shift Additions	Shifts after REDUCE
L99	99201	21025	7	7076
P068	87620	10485	5	6141
READ	59971	15814	7	2969
UBAG	98351	19926	10	2248

Table 2: Shift Selection in Column Generation

resulting in a need to resort to decomposition. Performance improvements in the mathematical programming component, in particular, alleviate this need by allowing the solution of larger problems. The most recent of these improvements, the column generation approach, has been demonstrated to give better quality schedules in acceptable time. A parameter driven approach gives the user more flexibility in solving problems using different solution techniques and allows different types and sizes of problems to be addressed.

References

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

[2] *Seventh International Workshop on Computer-Aided Scheduling of Public Transport Preprints*, Center for Transportation Studies, Massachusetts Institute of Technology, 1997.

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

[4] A. S. K. Kwan, R. S. K. Kwan, M. E. Parker and A. Wren, *Producing Train Driver Shifts by Computer*, Computers in Railways V, J. Allan et al (editors), Volume 1: Railway Systems and Management, Computational Mechanics Publications, pp 421-435, 1996.

[5] H. D. Sherali, *Equivalent Weights for Lexicographic Multi-Objective Programs: Characterizations and Computations*, European Journal of Operations Research, 18, pp 57-61, 1982.

[6] M. Desrochers, J. Gilbert, M. Sauvé and F. Soumis, *CREW-OPT: Subproblem Modelling in a Column Generation Approach to Urban Crew Scheduling*, Computer-Aided Transit Scheduling, M. Desrochers and J.-M. Rousseau (editors), Springer-Verlag, pp 395-406, 1992.

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

[8] J. J. Forrest and D. Goldfarb, *Steepest Edge Simplex Algorithms for Linear Programming*, Mathematical Programming, 57, pp 341-374, 1992.

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

[10] S. Fores *Column Generation Approaches to Bus Driver Scheduling*, PhD Thesis, University of Leeds, 1996.