Scheduling Train Crews: a case study for the Dutch Railways

RICHARD FRELING[1, 2], RAMON M. LENTINK[2], AND MICHIEL A. ODIJK[2]

[1] *Econometric Institute, Erasmus University Rotterdam (The Netherlands)*

[2] *ORTEC Consultants b.v., Gouda (The Netherlands)*

e-mail: freling@few.eur.nl

Econometric Institute Report EI2000-17/A

***Abstract*:** In this paper the problem of scheduling train crew is considered. We discuss a general framework of which the method for solving the train crew scheduling problem is a special case. In particular, our method is a heuristic branch-and-price algorithm suitable for large scale crew scheduling problems. This algorithm is applied to a real life train guard scheduling problem which is provided to us by the Dutch Railways. Computational results show that our algorithm is capable of getting sub-optimal solutions for a large scale instance within reasonable computation time.

Introduction

In this paper we consider a case study dealing with the scheduling of train crews at Dutch Railways (NS). The Crew Scheduling Problem (CSP) is a well-known problem in the field of Operations Research. It consists of designing duties using pre-defined tasks with fixed starting and ending times and locations. Each duty must satisfy a set of work laws and agreements. The objective is to minimize a combination of fixed costs (the number of duties) and variable costs (e.g. penalties for less desired constructions).
For many years public transport companies use planning systems which incorporate automatic crew scheduling. Although usually the crew scheduling problem is solved after the vehicle schedules have been designed, it has recently also become possible to design integrated vehicle and crew schedules for small to medium-scale problems (see the paper by Freling, Huisman and Wagelmans, presented at this workshop). After solving the crew scheduling problem, the

resulting duties must be assigned to individual crew members. This process is called crew rostering.

In this paper, we discuss a heuristic algorithm for the crew scheduling problem, which uses column generation to solve linear programming problems and a branch-and-price heuristic to get integer solutions. Columns are generated implicitly using a dynamic programming algorithm. The main benefit of this work is that it fits in a general framework, which can easily be applied to crew scheduling problems in different contexts. In fact, slightly different versions of this algorithm have been used for scheduling bus drivers and airline crew, and also for train and airline crew rostering. Furthermore, the approach allows for solving large scale crew scheduling problems within reasonable computation time, and is very robust in the sense that different scenario's with respect to for example union and/or governmental regulations can easily be implemented. The mathematical techniques are not discussed in great detail here. For a more comprehensive description of the subject, including the technical details, we refer the readers to Lentink (1999).

The heuristic algorithm has been tested in a case study, which deals with the scheduling of train guards. This study has been carried out for NS by ORTEC Consultants in the Netherlands. NS currently plan their crew manually, but recently several researchers have done case studies to see whether automatic crew scheduling is an interesting alternative (see also the paper by Fischetti and Kroon, presented at this workshop).

Most of the Operations Research literature on crew scheduling deals with bus driver scheduling and airline crew scheduling. From our experience, the main differences between scheduling train crew and bus drivers are that:

- train crew travels more often as passenger on a train to get to different locations,
- train crew can begin/end a duty at a relative longer distance from their home bases (sometimes spending the night at a hotel away from the home base),
- delays are more critical for trains because of the stronger interconnectivity of a train network.

Just like in the airline context, sometimes trains may be run all round the clock, which makes it harder to schedule the crew on a daily basis. In the airline context, a schedule (also called a pairing) usually may cover two or three days.

This paper is organized as follows. In the next section, we introduce the general framework that is used in a crew planning system at ORTEC Consultants for both crew scheduling and rostering problems arising in airline and railway applications. In Section 3, we introduce the case provided to us by NS. One of our primary research objectives for this particular case was to adapt our algorithms to be able to deal with large-scale problems. The resulting branch-and-price heuristic, which is a special case of the general algorithm, is presented in Section 4, where we also discuss techniques for speeding up the algorithm. Finally, in Section 5, we show computational results for the scheduling of train guards. For the instance provided to us by NS, which consists of 1114 tasks, we are able to get a sub-optimal solution within reasonable computation time.

Automatic Crew Planning at ORTEC

ORTEC Consultants is a medium sized company located in the Netherlands with about 250 employees. Since its foundation in 1981, ORTEC has worked in close collaboration with the Erasmus University Rotterdam. The company is specialised in decision support systems that contain (advanced) Operations Research techniques. More information about some systems that are focused on railway planning can be found in Lentink et al. (2000). Several personnel planning systems are developed at ORTEC for various fields of industry. One such system is CDR-Lite (Crew Duty Rostering), a crew planning system for the airline and railway industry. The module for automatic crew scheduling and rostering is based on a branch-and-price algorithm for a generalised set partitioning model. In this section, we briefly introduce this model and the general algorithm. In Section 4, we present a special version of this algorithm, which is the heuristic branch-and-price algorithm we used for the railway crew scheduling problem.

General Mathematical Formulation

The generalised set covering model presented below can be used to formulate most crew planning problems arising in practice. For similar work on a more general framework in the context of time constrained routing and scheduling problems, we refer to Desrosiers et al. (1995). These types of problems have in common that the column generation technique is used because the mathematical formulations contain a huge number of variables.

For presenting the model, we need the following definitions:

- R = the set of all feasible columns (e.g. duties);
- K = the set of all resource units (e.g. crew);
- I = the set of rows which need to be covered (e.g. tasks);
- $Q(i)$ = the set of columns covering row $i \in I$;
- $R(k)$ = the set of eligible columns for resource unit $k \in K$;
- d_r = the cost of column $r \in R$;
- p_i = a penalty for not assigning row $i \in I$.
- b_i = number of columns which need to cover row $i \in I$.
- c_k = maximum number of columns covering resource unit $k \in K$.
- X_r = 1, if column r is selected, 0 otherwise;

- S_i = 1, if row i remains unassigned, 0 otherwise.

The generalised set partitioning model on which our framework is based is as follows:

Minimise $\sum_{r\in R} d_r \; X_r \; + \; \sum_{i\in I} p_i \; S_i$

subject to

$$\sum_{r\in Q(i)} X_r + S_i = b_i \qquad \forall\, i\in I \qquad \text{(i)}$$

$$\sum_{r\in R(k)} X_r \; \leq \; c_k \qquad \forall\, k\in K \qquad \text{(ii)}$$

$$X_r, S_i \; \in \{0,1\} \qquad \forall\, r\in R,\ \forall\, i\in I \qquad \text{(iii)}$$

The objective is intended to minimise the total costs of the partition, plus the total penalty costs of uncovered rows. Constraints (i) are generalised set partitioning constraints, which state that exactly b_i columns of the set $Q(i)$ are in the solution. Variables S_i are added to allow rows to remain uncovered by X variables, for example, a task may not be covered by any duty. Constraints (ii) guarantee that each resource unit can be assigned to at most c_k columns. We assume that the intersection of the sets $R(k)$ for all k is empty, that is, each column $r\in R$ is uniquely defined for one resource unit. In case of crew rostering this means for example that a set of rosters (columns) is uniquely defined for each crew member. Variations for both Constraints (i) and (ii) are possible by interchanging equality signs and inequality signs. Additional constraints may be added for modelling global constraints which deal with sets of columns (e.g. a maximum percentage of duties with duration above 9 hours).

General Solution Approach

The general solution approach consists of an exact branch-and-price algorithm for the generalised set partitioning model. Branch-and-price is a special application of branch-and-bound, where column generation is used to solve linear programming relaxations with a huge number of variables. See also Barnhart et al. (1998) for a general discussion of column generation in the context of integer programming. In the last decade many papers, also presented in this and previous proceedings, deal with column generation approaches for public transport scheduling problems. Here, we only briefly discuss our approach. The main elements of the general solution approach are visualized in Figure 1. The column generation algorithm to solve the linear programming (LP) relaxation in the root node of the branch-and-bound tree starts with an initial selection of columns. Then, additional columns are generated implicitly while needed until the linear programming (LP) relaxation is solved to optimality. Rules that define the feasibility of a column are checked during the generation of columns. The procedure to solve the integer programming (IP) problem uses the same column generation algorithm to solve the LP relaxations in each other node of the branch-and-bound tree. Here, the column generation procedure is started with the columns in the final LP problem of the parent node.

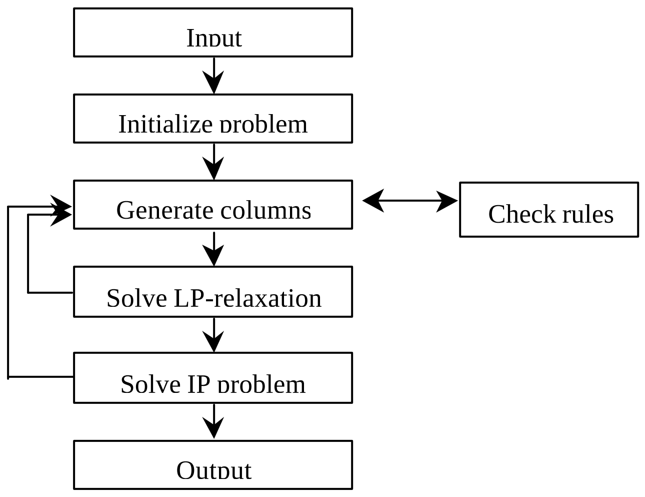

Figure 1: Flowchart of the general solution approach

Two aspects are of major importance in this solution approach: how columns are generated, and which branching rule is used. The system contains several algorithms for generating columns based on one or more acyclic networks. For example, for crew scheduling one network is used, while for crew rostering a network is used for each crew member. The networks are defined such that each node corresponds to a task (or trip, flight, duty, etc.) and each arc corresponds to a feasible sequence of two tasks in one duty. In addition a source and sink arc are added to each network, denoting the start and end of a duty. A path in the network corresponds to a feasible duty if the duty constraints are satisfied. See also Desrochers et al. (1992) for an example of a similar network for crew scheduling. Paths in the networks are constructed by solving a resource constraint shortest path algorithm (see Desrosiers et al. (1995)). The algorithms we implemented are dynamic programming, depth-first search, and all-pairs shortest path algorithms (see Freling (1997)). The choice of the algorithm depends on the application considered.

For the branching rule we implemented we branch on the arcs in the underlying networks (see e.g. Desrosiers et al. (1995)). Each branch consists of either forbidding or forcing one arc to be in the solution. The consequence in both nodes resulting from a branch is that several variables are fixed to zero, and that the corresponding networks are adjusted by deleting arcs to force or forbid an arc to be in the solution. In case of forcing an arc to be in the solution, all other arcs leaving and entering the corresponding nodes are deleted.

This general framework is very robust in the sense that both the constraints defining the feasibility of each duty, and the constraints defining the feasibility of a set of columns can be easily incorporated without changing the structure of the algorithm. In addition, for large scale problems several exact and heuristic techniques can be used in a straightforward manner to speed-up the algorithm (see Section 4.2).

A Case Study for NS

In this section we discuss a case study carried out for NS. The case consists of four Intercity lines. These lines connect the north east of the Netherlands with the urban part in the west, and therefore it is called the North-East case. The network representing these lines is visualised in Figure 2, which is taken from Fischetti and Kroon (1999).

Train guards and drivers have to be assigned to the trains that operate on these lines. For this case study, we examined the generation of schedules for guards on a weekday for intercity lines. The generation of schedules for drivers is outside the scope of this case. It is assumed that a timetable has already been generated, and also that rolling stock has been assigned to operate the timetable.

A task is defined as the minimum portion of work that has to be carried out by one person. Here, this means a part of a line where it is not allowed for a guard to change trains in between. The data consisted of 1114 tasks that had to be assigned to the guards. These tasks where not spread evenly across a day. In the morning and evening peaks more guards are necessary because more and longer trains are used. When the length of a train exceeds certain thresholds more guards are necessary. Figure 3 shows the demand for guards during the whole day.

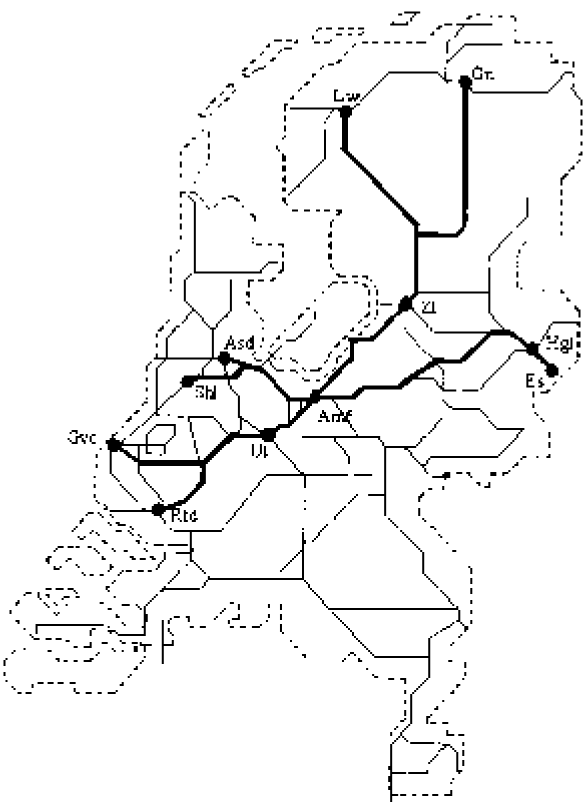

Figure 2: Lines that form the North-East case

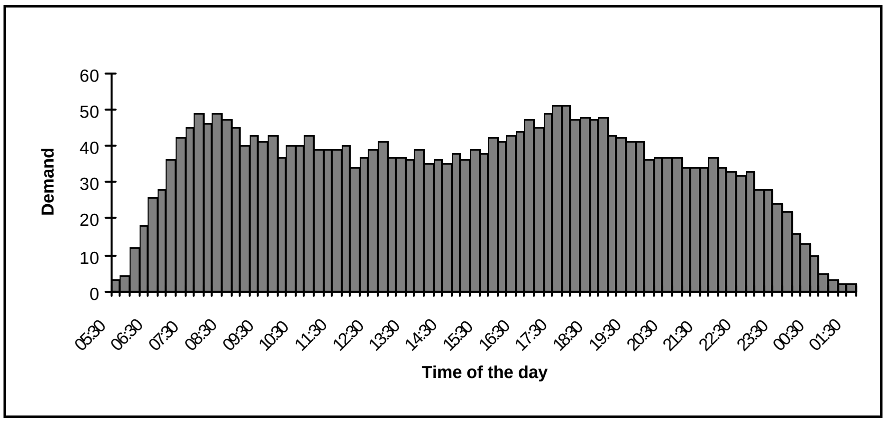

Figure 3: Demand for guards during a weekday

The duties must satisfy the following criteria:

- Efficient, that is, the number of required guards is as low as possible.
- Robust, that is, the duties must be of high quality with respect to delays of trains. This is taken care of by penalizing changes from one train to another within a duty.

The duties that are generated in the solution approach have to meet certain work laws and agreements in order to be feasible. These laws and agreements are described below. There are three levels of feasibility restrictions:

1. *High level restrictions*. Here we find rules that are called coupling restrictions because they relate to a set of duties. There are three of those rules in this case:

   a) At most 5% of all selected duties have duration shorter than 5 hours.

   b) At most 5% of all selected duties have duration longer than 9 hours.

   c) The average length of the selected duties is at most 8 hours.

2) *Medium level restrictions*. Rules at this level determine the feasibility of a duty. Most restrictions in the case are at this level:

   a) The maximum length of a duty is 9½ hours.

   b) The minimum length of a duty is 4 hours.

   c) A duty with a length of at least 5½ hours should contain a meal break of at least 30 minutes.

   d) The meal break should not start more than 5½ hours after the start of the duty.

   e) The meal break should not end more than 5½ hours before the end of the duty.

   f) Each duty must start with a sign-in time of 20 minutes.

   g) Each duty must end with a sing-off time of 15 minutes.

   h) A duty has to start and end at the same location.

3) *Low level restrictions*. A restriction on this level determines if two tasks can be performed consecutively:

   a) The end location of the first task and the start location of the second task should be identical.

   b) The second task can only start after the first task has finished.

   c) In case of changing of trains a buffer of 16 minutes should be taken into account.

In the next Section we discuss how these restrictions are incorporated in our solution approach.

A Heuristic Algorithm for the Railway Crew Scheduling Problem

The railway crew scheduling problem is formulated as a set covering model, which is a special case of the model presented in section 2.1. Constraints (ii) are left out because all resource units (duties) have identical characteristics. Furthermore, the S_i variables are not needed because we have unlimited resources, and we use a set covering version, that is, Constraints (i) have greater or equal sign. In fact, we need to use set covering because over-covers have a meaning in this context. Whenever a train crew needs to be positioned it is travelling as a passenger on a scheduled train. Thus, if a trip needs to be covered once but is covered twice, this means that one of the duties covering the trip contains a positioning. The right-hand-side of Constraints (i) may be larger than one, because more than one crew member may need to cover a trip. The resulting model is a generalized set covering model with side constraints. The side constraints correspond to the high level constraints dealing with sets of duties. Low level constraints are dealt with in the construction of the network, while medium level constraints are dealt with in the construction of paths during the column generation procedure. In the remainder of this section we present the heuristic algorithm and some acceleration techniques.

Heuristic Branch-and-Price

We did not use our general branch-and-price algorithm (see Section 2.2) directly, because of the large scale of the problems to be solved. Furthermore, it is not straightforward to use the special branching rule for the set covering formulation. Therefore, we modified the general algorithm into a heuristic branch-and-price algorithm. The column generation algorithm for solving the linear programming relaxation in the root node is identical to the general algorithm. However, for the branch-and-price procedure, we branch on the X_r variables, where we only consider the 1-branch, that is, we only solve the linear programming relaxation after fixing a variable X_r to one. As in the general algorithm, we use column generation to solve the linear programming relaxation in each node of the branch-and-bound tree. This branching rule can be used here with column generation because we only solve the 1-branch and stop once an integer solution is found. It would not be possible to solve the 0-branch because we can not forbid a column to be generated in our column generation algorithm.

Generating Duties using Dynamic Programming

We used dynamic programming for the generation of duties (see e.g. Desrosiers (1995) and Freling (1997)). For the column generation algorithm we need to find one or more paths that correspond to one ore more columns with negative reduced cost. Let D_s and D_l be binary parameters, where D_s equals 1 if a duty has duration shorter than 5 hours and 0 otherwise, and D_l equals 1 if a duty has duration longer than 9 hours. Furthermore, let m be the number of constraints in Constrains (i), let y_i be the dual variable corresponding to constraint i, and let y_{m+1}, y_{m+2} and y_{m+3} be the dual variable corresponding to the coupling constraints defined as high level restrictions 2a, 2b and 2c, respectively. Then, with t_r the duration of duty r in minutes, the reduced costs of a column r in the set covering model is defined as follows:

$$\bar{c}_{\mathrm{r}} = c_r - \sum_{j \in J_i} y_j + 0.95 * \left(D_s * y_{m+1} + D_l * y_{m+2}\right) + \\ 0.05 * \left((1 - D_s) * y_{m+1} + (1 - D_l) * y_{m+2}\right) + (t_r - 480) * y_{m+3}$$

Reduced costs need to be negative because we are minimizing. A feasible path up to a node in the network (except for the sink) is called a *partial path*. We can define the cost of arcs and a cost function for modifying the cost when adding an arc to a partial path. In this way the cost of a path from the source to the sink is such that it corresponds to the reduced cost of the corresponding column (duty) in the model. Similar, each constraint that defines the feasibility of a column can be translated to consumption on the arcs and a function for defining this constraint's consumption of a partial path when adding an arc to it. The reduced costs and the constraints are called *resources* (not to be confused with resources as defined in Section 2.1), and the problem of finding the minimum cost path satisfying the constraints is called the resource constraint shortest path problem.

For the dynamic programming algorithm, dominance tests are carried out at each node in the network in order to reduce the state space. That is, keeping only those paths that can possibly be part of an optimal solution reduces the number of feasible paths up to a node. For this case three resources can establish dominance. The first resource is the duration of a duty after the meal break. The smaller the duration of the partial path, the more time is left for tasks that can be performed from here (according to rule 2e). The second resource is the reduced costs of the partial path, that is, the lower the reduced costs of a partial path the better. The last resource is more complicated. This resource corresponds to the duration of (the part of) the duty corresponding to the partial path, where shorter duties are better. Because a duty should have duration of at least 4 hours (see rule 2b), this resource should only be used if the length of both paths is at least 4 hours. However, the coupling constraint (rule 1a) plays a role in this as well. Let partial paths p_1 and p_2 have the same value for the first two resources. Path p_1 has a duration of 4½ hours and path p_2 has a duration of 7 hours, so path p_1 would dominate path p_2. But because at most 5% of the selected duties may have duration of less than 5 hours, it may be that one ore more complete paths constructed from partial path p_1 will be dominated by complete paths constructed from partial path p_2 due to a change in reduced costs. This because the dual variables with respect to this coupling constraint can only be taken into account once a path is completed. Note that once the partial path has duration of at least 5 hours we know that this dual variable needs not be taken into account. The other two coupling constraints do not influence this criterion since the path on the 'safe side' is preferred.

Acceleration Techniques

In order to be able to solve the large scale scheduling of train guards, we used several acceleration techniques to speed up our algorithm. In Section 0 we present some computational results which will show some effects of these techniques on the computation time.

Reduced network size

We implemented several techniques for reducing the size of the network. As noted before, the number of guards required on a train depends on the length of the train among others. When more then one guard is necessary on a task there are two possibilities with respect to the mathematical model. It is possible to see the tasks as different and insert them separately in the network. This is a straightforward manner but is undesired because tasks that have to be performed by more then one guard can be found regularly. This means an extra side constraint in the formulation of section 0 but also extra nodes in the network with extra incoming and outgoing arcs. Therefore another approach was implemented, where the task is seen as the same and this task has to be covered more than once. As a result, the number of nodes as well as the number of arcs in the network decreases significantly because 'duplicates' are moved.

We briefly discuss three other network reduction techniques we used (see Lentink (1999) for a more elaborate discussion of these techniques):

- Remove arcs between the source/sink and nodes for which it is not allowed to start/end a duty. This is possible due to restriction on the location or on the start/end time of a duty.
- Remove arcs for which corresponding duration exceeds a value, which is based on the maximum duty duration

The results of the network reductions are shown in the next table. The last row contains the size of the final network that is used in the algorithm.

Rule	Nodes	Arcs
Original network without reduction	1,114	84,497
Removing duplicate arcs and nodes in the network	820	38,910
Further network reduction	820	26,513

Table 1: Results of network reduction.

We see that it is possible to reduce the network considerably, that is, the number of nodes can be reduced with more than 25%, while the number of arcs by almost 70%. These type of techniques are very important in order to solve large scale crew scheduling problems with our algorithm within reasonable computation time.

Dynamic network size

Especially in the first iterations of the column generation procedure a lot of paths can be generated with negative reduced cost, although we are interested in a few of them. Therefore, we can use a heuristic technique to speed up the procedure. This technique is focused on restricting the number of outgoing arcs at a node that will be considered. This number is updated dynamically during the column generation procedure. When tailing off (only small improvements in the objective function) is detected the number of outgoing arcs is increased. This continues until all outgoing arcs at all nodes are considered, so that optimality of solving the LP relaxation can still be guaranteed. The tailing off detection and the increase in number of outgoing arcs are the most important parameters for this technique.

Multiple variable fixing

An acceleration technique for the branch-and-price part of our algorithm is multiple variable fixing. Instead of fixing only one variable X_r to one in each branch, we fix all variables with fractional value above 0.6 to one (see also Gamache et al. (1999)).

Computational Results and Conclusion

In the table below we summarize the computational results for the case described in Section 3. This case consists of 1114 tasks that need to be assigned to duties for train guards. We tested the algorithm presented in Section 4 and some of its variations. The second to fourth columns in the table show the results for the basic algorithm including the network reductions (Section 4.3.1), and the fifth column without network reductions. The third and fourth column show the results for multiple variable fixing (Section 4.3.3), and dynamic network size (Section 4.3.2), respectively. The results shown in each column are, respectively, the CPU runtime for solving the LP relaxation in the root node (in seconds on a Pentium II/400Mhz/128Mb), the total CPU runtime of the algorithm, the objective value of the LP relaxation in the root node, the objective value of the final integer solution, the gap between these two, the number of branches needed to find the integer solution, and information about the solution: the number of duties, the average duration of the duties, the number of positionings, and the percentage of short and long duties.

	Method			
	Basic	Multiple variable fixing	Dynamic network size	No network reduction
CPU LP	225	225	228	1193
CPU total	1935	687	1978	11473
LP value	31064	31064	31064	31086
best integer	31801	32255	31801	34427
gap %	2.32	3.69	2.32	9.71
number of branches	102	14	102	102
duties	132	134	132	143
average duration	7:58	7:54	7:58	7:55
positionings	73	81	73	154
short duties %	1.52	2.24	1.52	3.50
long duties %	3.79	2.99	3.79	4.20

Table 2: Computational results

These results are a summary of the results presented in Lentink (1999). The cost function is a combination of fixed cost per duty, and variable costs for positionings and change of trains. We have imposed the condition that at most one change of trains is allowed in a duty. As can be seen from the table, the basic algorithm (including network reductions) is able to find a good solution with 2.32% gap within 32 minutes. Multiple variable fixing greatly improves the runtime to only a little more than 11 minutes but the gap increases to 3.69%. This means two extra duties and 8 extra positionings, but a decrease in average duty duration of 4 minutes. The dynamic network size did not perform well because of an extra runtime of 43 seconds. In previous research we noted that the dynamic network size can significantly improve the runtime when the network is larger. Finally, in the last column we can see that the runtime explodes to more than 3 hours when the network reductions are not used.

Naturally, although the results are very promising, we need to be careful when drawing conclusions based on one instance only. Therefore, future research is necessary to test the algorithm and its variations on other instances as well. Although we know of other research on similar cases provided by NS, we do not have access to information to provide a good comparison. In Kroon and Fischetti (1999) results are presented using the TURNI for the same case but with slightly different constraints. Their approach is based on column generation as well, but using Lagrangian relaxations and heuristics instead of LP relaxations and branch-and-price (see also Caprara et al. (1999)). The algorithm finds a first good solution within a few minutes (on a Pentium II/450Mhz/128Mb), while if the time is extended to 60 minutes only marginal improvements can be obtained. As mentioned before, the main advantage of our solution approach is its robustness. If some rules change this can be incorporated very easily in our approach and the algorithm needs very little tuning.

References

Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance (1998). Branch-and-Price: Column Generation for Solving Huge Integer Programs. Operations Research 46, pp. 316-329.

Caprara, A., M. Fischetti, P.L. Guida, P. Toth, D. Vigo (1999). Solution to large scale railway crew planning problems: the Italian experience. In: N.H.M. Wilson (ed.): Computer-Aided Transit Scheduling: Proceedings of the Seventh International Workshop, Springer Verlag, Berling, pp. 1-18.

Desrosiers, J., Y. Dumas, M.M. Solomon, F. Soumis (1995). Time constrained routing and scheduling. In: M.O. Ball/T.L. Magnanti/C.L. Monma/G.L. Nemhauser (eds.): Handbooks in Operations Research and Management Science 8: Network Routing, North-Holland, Amsterdam, pp. 35-139.

Desrochers, M., J. Gilbert, M. Sauve, F. Soumis (1992). Subproblem modeling in a column generation approach to the urban crew scheduling problem. In: M. Desrochers and J.M. Rousseau (eds.): Computer-Aided Transit Scheduling: Proceedings of the Fifth International Workshop, Springer Verlag, Berling, pp. 395-405.

Fischetti, M., L. Kroon (1999). Scheduling Train Drivers and Guards: The Dutch "Noord-Oost" case, working paper. Management Report no. 25-1999, Erasmus University Rotterdam.

Fischetti, M., L. Kroon (2000). Solving large-scale crew scheduling problems at the Dutch railways. Paper presented at this workshop.

Freling, R. (1997). Models and techniques for integrating vehicle and crew scheduling. PhD Thesis, Tinbergen Institute, Erasmus University, Rotterdam.

Freling, R., D. Huisman, A.P.M. Wagelmans (2000). Applying an integrated approach to vehicle and crew scheduling in practice. Paper presented at this workshop.

Gamache, M., F. Soumis, G. Marquis, J. Desrosiers (1999). A column generation approach for large-scale aircrew rostering problems. Operations Research 47, pp. 247-263.

Lentink, R.M. (1999). Crew Scheduling voor NS Reizigers. Master's thesis, Free University Amsterdam (in Dutch).

Lentink, R.M., M.A. Odijk, R. Freling, J.S. de Wit (2000). Making Operations Research techniques work to facilitate and improve railway planning. Paper to be presented at CompRail 2000, Italy.