Branch-and-Price Algorithm for a Real-life Bus Crew Scheduling Problem

Sylvain FOURNIER

[1]WPLEX Software
Rod SC 401, 8600 Corporate Park bloco 5 sala 101
88050-000 Santo Antônio de Lisboa, Florianópolis SC

sylvain@wplex.com.br

Abstract. *In this paper, we describe a Crew Scheduling Problem faced by bus transit companies, whose main feature is the possibility of various workday types, each of which has its own bounds on the durations of each part of the workday. The problem is solved using a classical Branch-and-Price algorithm. The huge size of the instances to be solved require some acceleration techniques that we describe briefly here.*

1. Introduction

WPLEX Software dedicates itself to the improvement of productivity in urban and air passenger transport, using advanced information technology techniques. One of its softwares, WplexON, allows transit companies to visualize and schedule their fleet and crew. These companies have to deal with some working rules imposed by legislation, in addition to their own internal regulation. When generating the drivers' workdays, they wish to minimize the costs comprised by salaries the workforce and overtime. To help these companies, WplexON provides a tool for the generation of the drivers' workdays out of the bus schedules. The software has to consider all the features of the problem and proposes a set of valid workdays that covers the bus schedule with a low total cost.

Section 2 describes the Crew Scheduling Problem faced by these companies. A general insight of the Branch-and-Price solving technique is given in section 3. In section 4, we provide a more detailed description on the generation of new workdays in the subproblem. The results are presented and analysed in section 5. Conclusions and perspectives for future work are given in section 6.

2. A Real-life Crew Scheduling Problem

Given the buses schedule for the day, forming subsets of trips usually called **blocks**, the aim is to create valid workdays with respect to the working rules, covering all the **tasks** (that can be trips or deadheads) of the bus schedule, and minimizing the salaries and costs of overtime. This kind of problem is known as a **Crew Scheduling Problem** (CSP).

Here, the legal rules are gathered into possible **workday types**. In a typical workday, the driver has a meal break, which can be paid or not, and divides the working day into two pieces or parts. Each workday type has upper and lower bounds on the working duration of each workday part (first part, break, second part), as well as on the total working time. In the case the break time is paid, the break duration has to be added to the total working time.

The cost of a workday is a piecewise linear function of the total working time of the driver: if this total time exceeds a given limit (that depends on the workday type), overtime has to be paid with a penalty over the regular time cost.

When a driver starts a part of its workday (beginning of his workday or after his break), he has to perform some action with the vehicle (checking, preparing the bus) before being able to use it. This action is called **briefing**, and usually requires some time, as well as the reverse action, the **debriefing**, that takes place at the end of a part of the driver's workday. The only exception where these actions are not performed by the driver is when he stays in the same bus before and after his break. In this case, there is neither debriefing before the break, nor briefing after it.

Finally, every workday part (before or after the break) must contain at least one trip (and not only deadheads).

3. Mathematical modeling based on a Branch-and-Price algorithm

Due to the nonlinear complex costs and the various workday types, a **set-partitionning** formulation was chosen to model the CSP, and a **Branch-and-Price** algorithm to solve this formulation. This technique is well-known and has been widely used to solve the CSP [Barnhart et al. 1998]. Let T be the set of tasks to perform. Let $I(t)$ be the set of all workdays covering task $t \in T$, and I be the set of all workdays. The binary variable x_i equals 1 if and only if workday i is part of the solution, and c_i is the cost of this workday. The CSP can be formulated as follows:

$$\min \sum_{i \in I} c_i \cdot x_i \tag{1}$$

$$\sum_{i \in I(t)} x_i = 1 \quad , \quad \forall t \in T \tag{2}$$

$$x_i \in \{0,1\} \quad , \quad \forall i \in I \tag{3}$$

Note that the workday types do not appear in this formulation as are included in the definition (feasibility) of the workdays. Another advantage of this formulation is that the cost on workdays doesn't need to be decomposed over tasks. The problem of the large number of variables is resolved by solving the problem on a small subset of all possible workdays, and adding gradually other promising workdays.

In many cases [Desrochers and Soumis 1989, Abbink et al. 2007] a set-covering formulation (with a '$\geq$' sign replacing the equality in constraint (2)) is prefered as its linear relaxation is easier to solve than that of the set-partitioning formulation. Here, it is impossible to use the set-covering formulation because of the constraints on the minimal duration. Indeed, the minimal duration constraints may prevent some sub-workdays of any feasible workday from being feasible. Consequently, the over-covering of tasks that may occur using a set-covering model can't be resolved by introducing sub-workdays in the solution.

For our branching strategy, we use the method described in the general case by [Barnhart et al. 1998] and detailed for the particular case of the Crew Scheduling Problem by [Desrochers and Soumis 1989]. The idea is to branch on pairs of consecutive tasks u and v, such that:

$$0 < \sum_{i \in I(u) \cap I(v)} x_i < 1 \tag{4}$$

The main advantage of this scheme is that on each branch ($\sum_{i \in I(u) \cap I(v)} x_i = 0$ or 1), the subproblem of generating new workdays is simplified by removing arcs or nodes in a graph.

Many columns are generated at each node of the Branch-and-Bound tree (as suggested by [Desaulniers et al. 1999]) and pricing is done until no new workday with negative cost can be added. At this point, if the solution is binary the algorithm stops. Otherwise a new branching is triggered.

Note that the aim here is to solve the problem in reasonable time, for instances whose size can reach a thousand tasks. Therefore, our interest is not to check whether the algorithm is optimal, but rather to find a good solution in little time. For this purpose, when solving the largest instances it is necessary to introduce acceleration schemes. For example, only one column generation is performed at each node of the Branch-and-Bound tree, except at the root where the columns are generated until no new workday with negative marginal cost can be found. The column generation process can also be stopped if the solution hasn't been improved enough for a given number of iterations.

The first set of columns in the master problem is made of "**trippers**", that is, small workdays (generally containing one single task) that are not necessarily valid. In order to avoid the cases where the linear relaxation of the master problem has no solution (it may occur during the branching), the trippers are always kept in the variables of the master problem, even when they have a fractional value and should be removed by branching.

4. Generation of new workdays

To generate a new set of workdays to enter the variables of the master problem, we use the technique described by [Desrochers et al. 1992]. The same kind of graphs was used for a scheduling problem by [Lopes and de Carvalho 2007] and for a cutting stock problem by [Alves and de Carvalho 2008]. Here, a graph is created from the bus schedule. The definition of the graph is illustrated for a small example on Figure 1. There are two blocks, the first one including two tasks (Task 1 and Task 2) and the second one only one task (Task 3).

Each task $t \in T$ is associated with two nodes in the graph: one (s_t) for the beginning and another (e_t) for the end of the task. In our case, it is necessary to introduce two more nodes per task $t \in T$: s_t^- before the briefing and e_t^+ after the debriefing. In addition, two nodes s (source) and e (sink) are used as extremities of the graph. The arcs can be defined as follows:

- Arcs of tasks between two nodes ($s_{t_1} e_{t_2}$) into the same block, which stands for performing all the tasks included between tasks t_1 and t_2,
- Arcs of break between two nodes from distinct blocks ($e_{t_1}^+ s_{t_2}^-$), which stands for a break after performing task t_1 and before task t_2,
- Arcs of break between two nodes from the same block ($e_{t_1} s_{t_2}$) as in this case, the briefing and debriefing don't have to be performed,
- Arcs of sign-on (between s and $s_t^-, \forall t \in T$) and sign-off (between $e_t^+, \forall t \in T$ and e),
- Arcs of briefing (between s_t^- and $s_t, \forall t \in T$) and debriefing (between e_t and $e_t^+, \forall t \in T$).

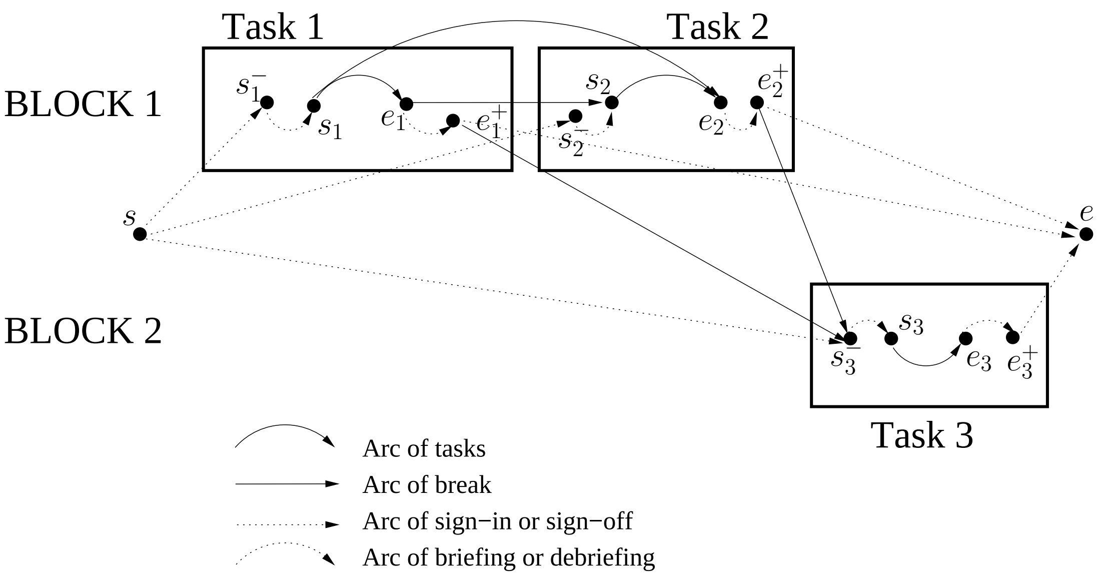

Figure 1. Graph for the subproblem

Any path in this graph can then be seen as a workday. In the example of Figure 1, path $(ss_1^- s_1 e_1 s_2 e_2 e_2^+ e)$ defines a workday with:

- a briefing followed by Task 1,
- a break between Task 1 and Task 2,
- Task 2 followed by a debriefing.

For any couple of tasks u and v in the same block, let $[u, v]$ be the set of all tasks to be performed between u and v (including u and v). For any task $t \in T$, let π_t be the dual cost of the partitioning constraint (of type (2)) associated with task t, after solving the linear relaxation of the master problem. At each step of the algorithm, the cost of each arc of tasks between tasks u and v (in the same block) is defined as follows:

$$c_{s_u e_v} = - \sum_{t \in [u,v]} \pi_t \tag{5}$$

The cost for all other arcs are set to 0. To obtain the next set of workdays to enter the set of variables, a shortest path problem with resource constraints is then solved using a dynamic programming algorithm. Indeed, constraints on the minimum and maximum duration of each part must be taken into account. As a result, in addition to the cost, the paid time has to be accumulated on partial paths during the processing of the algorithm.

To handle the case with several workday types (each one with its own constraints and costs), the generation of new workdays is divided into several subproblems, each one corresponding to a workday type. The drawback of having to solve more subproblems at each step of the algorithm is compensated by the fact that, for each workday type, the graph is not as big as if all workday types were gathered in the same graph, as some arcs can easily be removed with respect to the constraints of duration.

5. Computational results

The algorithm has been tested using the software GLPK 4.38 to solve the linear relaxation of the master problem, with an Intel Core 2 Duo 2.66 GHz, 4GB RAM.

Table 1. Comparison of the solution value (time in seconds)

Instance	With all the workdays			Generating workdays		
	Time	Cost	Steps	Time	Cost	Steps
I-79-6	565	342.13	3	49	342.13 (0%)	94
I-87-4	2043	368.77	1	25	368.77 (0%)	43
I-96-7	414	494.2	17	44	496.28 (0.42%)	124
I-68-6	81	377.92	23	25	381.78 (1.02%)	182
I-120-11	282	607.37	5	45	609.53 (0.36%)	103
I-111-11	374	559.78	19	82	559.78 (0%)	194
I-46-3	11	390.99	23	9	396.95 (1.52%)	107

We compare the solution value of the algorithm with the one obtained using the Branch-and-Bound algorithm alone, starting with all the possible workdays as variables of the master problem. Obviously, this comparison can be applied only for small instances. The tests are carried out on instances from WplexON customers. Each instance is denoted by "I-n-m" where n is the total number of tasks in the instance and m the number of blocks. Note that to establish the complexity of an instance, we may consider first the number of tasks as the more there are, the more nodes (and arcs) there are in the subproblem graphs. However, for a given number of tasks, the problem is more difficult if there are fewer blocks, as the subproblem graph then contains more arcs (more task arcs as well as more break arcs). As a result, an instance may be seen as "difficult" if it contains many tasks in few blocks.

The results are summed up in Table 1. For each run of the algorithm (with all the workdays at the beginning on the one hand, generating the workdays step by step on the other hand), the computing time (in seconds), the cost of the best solution found and the number of steps in the algorithm are shown. The number of steps is the total number of times the master problem is solved. In the cost of the solution obtained with column generation, the percentage given into brackets is the percentage of the difference with the cost of the solution obtained in the first case.

We can note that the cost of the solution with column generation is often the same as the cost of the solution obtained with all the workdays, and when it is not the case, the difference is low. The column generation is then very effective, and it should be pointed out that the processing time is far lower in the context of column generation. The difference in the number of steps can be explained by the absence of column generation in the first case. And for instance I-87-4, the first solution found is integer and there is no need in branching either.

The greatest instances are solved within reasonable time, leading to solutions WplexON customers are satisfied with. Instances with 300 tasks are usually solved in less than an hour, and instances with 1000 tasks can usually be solved in a few hours.

Unfortunately, the comparison of our approach with other methods of the literature is difficult because of the particular characteristics of our problem. For example, few papers deal with briefing and debriefing times when generating the crew schedule. Moreover, most of the other approaches solve the problem using a set-covering formulation, which is impossible here, as pointed out in section 3.

6. Conclusion and perspectives

The Branch-and-Price algorithm described in this paper is efficient and has allowed Wplex-ON to solve large customer instances. The main difficulties stand in the design of the subproblem, in particular the definition of the graph.

Our future research is mainly guided by the size of the customer instances. Indeed, we need to decrease the computing time for instances with up to thousands of tasks. Other improvements would be useful, such as the simplification of the subproblem graph (for example, removing the useless nodes), or providing an heuristic first solution at the beginning of the algorithm in order to get a good upper bound early. The fact that trippers are always maintained in the master problem could also be replaced by a treatment of the cases when the master problem gets no solution (using the infeasibilities as costs in the master problem, for example). Noticing that the same variables remain fractionary after a branching, one could also branch on several independent pairs of consecutive variables instead of one at a time.

References

[Abbink et al. 2007] Abbink, E., Wout, J. V., and Huisman, D. (2007). Solving large scale crew scheduling problems by using iterative partitioning. In Liebchen, C., Ahuja, R. K., and Mesa, J. A., editors, *ATMOS 2007 - 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany. Internationales Begegnungs und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. 2

[Alves and de Carvalho 2008] Alves, C. and de Carvalho, J. M. V. (2008). A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers and Operations Research*, 35(4):1315–1328. 3

[Barnhart et al. 1998] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., and Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329. 2

[Desaulniers et al. 1999] Desaulniers, G., Desrosiers, J., and Solomon, M. M. (1999). Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. Technical Report G-99-36, Les Cahiers du GERAD. 3

[Desrochers et al. 1992] Desrochers, M., Gilbert, J., Sauvé, M., and Soumis, F. (1992). Crew-opt: Subproblem modeling in a column generation approach to urban crew scheduling. In Desrochers and Rousseau, editors, *Computer-Aided Transit Scheduling*, pages 395–406. Springer-Verlag, Berlin. 3

[Desrochers and Soumis 1989] Desrochers, M. and Soumis, F. (1989). A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23(1):1–13. 2

[Lopes and de Carvalho 2007] Lopes, M. J. P. and de Carvalho, J. M. V. (2007). A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times. *European Journal of Operational Research*, 176(3):1508–1527. 3