# Train Routing Algorithms: Concepts, Design Choices, and Practical Considerations

Luzi Anderegg, Stephan Eidenbenz, Martin Gantenbein,
Christoph Stamm, David Scot Taylor,
Birgitta Weber, and Peter Widmayer

Institute of Theoretical Computer Science, ETH Zentrum,
CH-8092 Zürich, Switzerland
{anderegg|eidenben|mgantenb|stamm|
taylor|weberb|widmayer}@inf.ethz.ch

**Abstract.** Routing cars of trains for a given schedule is a problem that has been studied for a long time. The minimum number of cars to run a schedule can be found efficiently by means of flow algorithms. In practice, the objectives are more complex, with many cost components such as shunting or coupling/decoupling of trains, and also with a variety of constraints such as requirements for regular maintenance. Most realistic problems are $\mathcal{NP}$-hard, and thus the standard powerful tools (e.g. branch-and-bound, branch-and-cut, Lagrangian relaxation, gradient descent) have been used. These methods may guarantee good solutions, but not quick runtimes.
In this paper, we advocate and test a different approach. In reality, not all constraints or objectives can be easily formulated mathematically. To allow for interactive human modification of solutions, the system must work rapidly, allowing a user to save desired subsolutions, and modify (or just start over on) the rest. After careful examination to find which constraints and costs we easily integrate into a the flow model approach, we heuristically modify network flow based soltuions to account for remaining constraints. We present experimental results from this approach on real data from the German Railway (DB) and Swiss Federal Railway (SBB).

## 1 Introduction

Train companies have to face many algorithmically challenging questions. One problem with huge economic impact is to minimize the number of trains needed to fulfill a given schedule of train routes. This problem, known as the *fleet size*, *rolling stock rostering*, or *vehicle routing* problem, was modeled as a minimum cost circulation problem long ago by Dantzig and Fulkerson [DF54]. In practice, the problem is still solved by hand, because this flow model ignores many constraints and objectives caused by ugly realities such as shunting, coupling/decoupling, and maintenance constraints.

In this paper, we report on realistic rolling stock rostering, as it occurs for large train operators such as German Railways (DB) and Swiss Federal Railways (SBB). We consider the standard minimum cost flow model, but we more carefully address what different costs and constraints it can represent. A brief introduction to the problem (Section 1.2) and the network flow approach (Section 1.3) are followed by more lengthy discussion of real-life problem variants (Section 2). We consider additional requirements which can be easily by the model in Section 3, which includes more complex costs, different train types, and various station constraints. In Section 4, we address maintenance, which does not seem to easily fit into the standard flow model, and thus requires additional effort. We describe our heuristic modifications to the standard flow model to satisfy maintenance constraints. In Section 5, we describe our experiments and results using data provided by the German Railway (DB) and the Swiss Federal Railway (SBB).

### 1.1 Definitions

We begin with some basic definitions. A *train unit* is the smallest entity of a train which must be considered separately. It may consist of a locomotive or a single railcar, but may also consist of several cars, with or without a locomotive (*self-locomotive train units* have locomotives), which are liked together. Train units are atomic, routed and maintained as an entity, never to be broken into smaller units in the problems considered

here. A *train* consists of a number of train units , coupled together, where at least one unit is a self-locomotive train unit. The *length of a train* is the number of train units which form the train. (The physical length of the train, a distance measurement, is implicitly used in some of the constraints of train scheduling as well.) There can be different kinds of train units, we will call them *train unit types*. Thus, a train unit type is the class of real instances of train units of the same type. Finally, a if a locomotive train unit is bi-directional, it is called a *push-pull* train.

A *route* is the exactly specified track between two given end stations. On such a route there may be several intermediate stations which are not considered. A *ride* is the movement of a train on a route. Subject to different tasks of rides we distinguish between productive rides, piggy-backing rides, empty rides (deadheading movements), and maintenance rides which comprise movements to and from a workstation and the actual maintenance work. Piggy-backing rides are unused train units which are coupled to a productive ride. Usually, they are cheaper than empty rides, because of the saved crew costs.

### 1.2 Problem description

The goal of *rolling stock rostering* is to assign a set of real train units to a predetermined schedule of train routes. The input of the simplest version of this problem will only include the scheduled routes, just one train unit type, and some cost functions for using specific train units of that type. Then, the output will assign train units to those routes, and will determine how many train units should be sent on each route, which specific train units will perform each route, and which empty rides are used to move train units between stations.

A general goal is to minimize the cost of the assignment. Of course, realistic costs are determined by a long list of factors, including the total number of train units needed, how much mileage they cover, crews used, number of couplings and decouplings, etc. The number of train units used is clearly one of the most important costs, and has been the primary focus of much of previous work (see also [EGH$^+$01]).

Besides these basic requirements, there are numerous additional requirements and constraints that may differ for each train company. We will present these requirements in detail as they are for German Railway (DB) and Swiss Federal Railway (SBB).[1] Depending on the constraints considered, there are many versions of the problem, and these have rarely been treated in the literature. One of the most important constraints has to do with train maintenance: trains need regular inspections, maintenance work, and cleaning, all of which must be done at special facilities. Any solution without maintenance is not very useful to train companies.

### 1.3 Flow model

Even the simplest version of the rolling stock rostering problem is often broken into two separate phases: first, one can solve the *train length* problem, in which the length of the trains on each route (including empty rides) are determined. The main goal here is to make sure that there are enough train units available at all times at each station to satisfy all scheduled rides at that station. In this part rides are only assigned to train unit types, but not to specific train units.

In the second phase, the *train assignment* problem, specific train units are assigned to the scheduled rides for each station. Thus, given a train assignment, one can determine what the next movement is for a specific train unit, given its previous movement. So, the train assignment results in a (cyclic) route for each specific train unit. Such a route is called a *rotation*.

A standard (and longtime) approach (see [Sch93,BHR99]) for modeling the train length problem phase is to use a flow model: a periodic directed graph is used to model the scheduled routes. Each vertex in the graph represents one station at a specific time. A route is represented by an edge, leaving a vertex representing its departure (station and time), and entering a vertex for its arrival (station and time). The minimum flow over an edge represents the passenger demand for its associated route, while upper bound capacities represent track/station limitations and thus a maximum number of allowed train units. Additionally, edges are added to the graph to represent different train actions, e.g. trains waiting within a station after their arrival, or empty movements from one station to another.

We already mentioned that our graph is periodic. For a daily period, this means that we can consider *overnight* edges, edges which can represent some train action overnight. For instance, if a train route leaves

[1] We would like to express our sincere gratitude to DB and SBB for sharing their real-life problems.

station $x$ at 22:00 and arrives at station $y$ at 1:00, it is clear that the train has not traveled back in time, but that instead it arrives at $y$ the next day. (Long routes, in which the train arrives more than one period later, can also be modeled without problem.)

Generally, this model is used to solve the train length problem, with the flow on each edge representing the number of train units on that route each period. The periodicity of our graph model implies that such a solution can be repeated each period: each period will begin with the same number of train units in a station as the period before. We call such a solution a *circulation*. Train assignment is done by breaking the train length solution into cycles, or *rotations*, and assigning physical trains to each rotation. These rotations may overlap each other, or contain an edge more than once, for edges assigned flow more than one in the solution. For these "long train" edges, it is convenient to think of multiple edges, each of flow one, to distinguish among them (i.e. the first and second units of a train ride are considered as separate routes, although they travel together).

### 1.4 Model limitations

Ideally, the train length and assignment problems should both be solved at the same time. Because the train length problem does not assign tasks to specific train units, many specific constraints (e.g. coupling times, maintenance) cannot be considered. Thus, a solution to the train length problem may not have any legal train assignment solution, if all constraints are followed. For some constraints (such as coupling), we can modify our input somewhat (add coupling time to routes) to ensure that coupling will not effect the feasibility of a train assignment (of course if this padding time is not needed, it may cause the optimal solution to be overlooked). Starting with Section 2, we describe our system which finds feasible solutions on realistic data.

Space limits our description of possible edge types, but in the full version of the paper, we will describe how the standard edge type assumptions, and even assumptions of the form of the problem solution, have sometimes been incorrect, leading to "optimal" solutions with higher cost than the true, best solutions. See also [EGH+01,Gan01]. We will call these solutions *one period solutions*, and although they may actually have higher costs, train companies prefer them, perhaps because they are easier to think about. Their name does not imply that each of the rotations needs one period to complete, they can take much longer. (They do need *at least* one period.) What is implied is that a rotation which takes $x$ periods to complete also requires $x$ train units. On each day, a single unit will complete the rides for the $i$th day of the rotation, and will finish the day at the starting position for the $(i+1)$st day of the rotation (wrapped over the $x$ days). This implies that once any train unit consecutively performs two rides, then in every period those two rides will both be serviced by one train unit.

### 1.5 Previous work

For a survey on the literature of the standard minimum cost flow approach, see Desrosiers et al [DDSS95]. More recently, in [EGH+01], it is proven that it is $\mathcal{NP}$-hard to approximate the rolling stock rostering problem arbitrary closely. This holds even for solutions with highly simplified maintenance constraints, with all train lengths set to 1, and the costs equal to the number of trains needed. In [Gan01], a variant of a proof from [HS89] shows that the train length problem is also $\mathcal{NP}$-hard, if arbitrary fixed costs are allowed. In this case (the train length problem), maintenance is not even a consideration, but instead more complex costs are allowed. In general, every realistic problem variant is difficult, but here we concentrate more on getting good solutions to real problem instances.

## 2 Practical issues

This section describes in more depth some of the real world problems which arise in rostering. As in Section 1.2, the primary data for all problems is a set of scheduled train routes, and the output must specify a set of trains and how they will be used to fulfill the input schedule. However, unless the basic model is extended to include additional costs and constraints, the solutions it produces will be of little value to to railway companies like DB and SBB.

Clearly, some of these changes require additional input is needed: specific knowledge for each route (e.g. as length, traveling time, whether or not the track will accommodate electric locomotives) is important. For

every type of train unit, maintenance requirements and infrastructures, cost per kilometer to run the train unit, cost to maintain a train unit, crew costs, and coupling/decoupling costs must also be considered. In following subsections, we describe some of the most important problem variants, constraints, and costs which arose in discussions with DB and SBB.

### 2.1 Variants

Here, we list just 3 variants which the standard model currently ignores.

- *Non-identical train unit types:* Each scheduled ride must be carried out by a predefined set of possible train unit types. Typically, each ride can be carried out by three or four alternative train unit types. Any solution approach must assign a valid train unit type to each scheduled ride. It is, however, possible to append train units as piggy-back units to a scheduled ride, even if the piggy-back units are of invalid types for the scheduled ride, as long as there are enough valid units in the scheduled ride.
  The concept of train unit types being assigned to individual rides very elegantly models different parts of a train: if, for example, a train should leave station $A$ at time $t$ to travel to station $B$ and consist of a locomotive, three first-class cars, six second-class cars and a dining car, we would model this as four different rides all leaving station $A$ for station $B$ at time $t$ with the first ride only allowing locomotives as train unit type, the second ride allowing only three first-class cars as train unit type, the third allowing only six second-class cars, and the fourth being a dining car. In our graph model, this will result in a multi-edge with multiplicity 4.
- *Minimum station turn around times:* If a train reaches its arrival station, it might have to be decomposed into its train units by decoupling and shunting; the train units are subsequently combined into new trains to carry out their next rides. The time needed to complete this procedure depends on the train unit types involved, on whether only coupling or decoupling or both are needed, and on the station topology.
  The minimum times needed for this procedure are called minimum station turn around times. These are given as part of the input for each possible combination of coupling mode, train unit type and station. Any feasible solution must fulfill all these timing requirements.
- *orientation:* Using push-pull trains we pay attention to the orientation of this train. The orientation of a train can change between arriving and leaving of a station. This station does not have to be a final station of a scheduled route. The position of the locomotive should be the same for each pair of time and station for all days in a period

### 2.2 Maintenance requirements

A feasible rostering solution must satisfy all maintenance requirements. The input information concerning maintenance is made up of three parts:

- *Maintenance types:* Each train unit must have certain maintenance types performed on it. The most common types are: Refuel for diesel locomotives (T), Interior cleaning (I+E), Exterior cleaning (ARA), Scheduled repairs (INST), and Technical check-ups (V+A). If a train has to wait at a station for a longer period of time, sometimes also called siding (ABST), this is considered maintenance as well.
- *Maintenance interval:* Each maintenance type must be performed on a train unit at certain intervals. These intervals can depend on elapsed time (since last maintenance), distance driven (since last maintenance), or on both. For example, interior cleaning should be done every 24 hours, which is a time-dependent requirement; exterior cleaning should be done, whenever the train has run for at most 1000 kilometers, which is a distance dependent requirement; finally, a technical check-up needs to be performed, whenever either one week has passed or 1000 kilometers have been driven, which is a requirement that is both time- and distance-dependent. Of course, it is always possible to perform maintenance before the interval is reached.
- *Maintenance stations:* Maintenance stations are scattered rather scarcely all over the network. For each maintenance station we are given the information, which maintenance types can be performed for which train unit types at which hours at this particular maintenance station. Moreover, capacity constraints are also given as part of the input and it is specified how long each maintenance type takes.

### 2.3 Costs

Typically, railway companies have many "hazy" objectives concerning the rotations. For example, the train unit movement per period (measured in moving time and distance) should be more or less equal for all train units. This objective ensures an almost equal aging for all train units, but it does not imply that all rotations should be of the same period length. Another possible objective is the minimization of the number of different stations during one period or in one rotation. This may increase the chance that a specific train unit will move along the same line during one period and thus increase the regularity of a rotation.

More concretely, the overall objective of our problem is to minimize costs. Although many costs such as the ones above are desired, they are also not well defined by the railway companies. Here, we consider costs which are easier to define, which can be given as part of the input. We distinguish between three cost types (this list is not exhaustive):

1. *Fixed costs:* Fixed costs occur for each train unit that is used at some stage in the rostering. Fixed costs include all costs that are incurred by simply owning the train unit without riding it at all; these costs include several items, but depreciation is the most important one. Traditionally, fixed costs are considered to be the crucial cost block. The quality of a train rostering solution is very often measured simply by the number of train units used. Train units are very expensive, and it costs tens of thousands of dollars per year *even to maintain a train which is not used,* though this could also be counted as maintenance costs, below.
2. *Costs for each ride:* Each ride, whether scheduled, empty or piggy-back, incurs a cost that is composed of different cost factors. The most important cost factors are the following:
   - *Cost per distance:* Each train unit type has a certain cost associated to it for each kilometer that it runs. Cost examples include power or fuel usage.
   - *Cost per time:* Each train unit type has a certain cost associated to it for each hour that it runs. Cost examples include heating, lighting.
   - *Cost per ton kilometer:* Each train unit type has a certain cost associated to it for each ton that is transported a kilometer. Cost examples include the wear and tear of rails.

   These cost factors are different for each type of ride, i.e., different for scheduled, empty, or piggy-back rides, with the empty rides being cheaper than scheduled, but more expensive than piggy-back rides. Moreover, these costs can occur either for each train unit in a train or for a train as a whole. For example, cost per time must be paid for a train as a whole to reserve rails for the train's passage.
3. *Maintenance costs:* Whenever maintenance is carried out on a train unit, costs are incurred. These costs depend on the maintenance type and which maintenance station does the work.

## 3 Flow model modifications

Because of the numerous additional requirements that a real-life problem sets as detailed in the previous section, our straight-forward two-phase approach of first solving the train length problem by building a graph and finding a minimum cost flow on it and then solving the train assignment problem by extracting rotations from the minimum cost flow may no longer be a viable approach. Here, we consider some of the additional requirements which *can* be treated by the approach, by making minor modifications to the graph that we build up for the train length problem.

### 3.1 Multiple train unit types

We integrate the concept of having several train unit types by first determining an order in which the different train unit types will be processed. The order is such that car-only train unit types are before self-locomotives and also before locomitives; moreover, inexpensive train unit types come before more expensive train unit types.

So, given the train unit ordering, we simply iterate the standard flow model on the cheapest unit first, for all routes where that car type can be used. Once we then iterate on the remaining routes.

Many complexities are hidden here: for instance, if the number of a certain type of train unit is limited, perhaps there are not enough to cover all routes for which that train unit is useful. This will leave more routes

for the next iteration. Also, the number of non-locomotive trains on a route may determine how many, or what type, of locomotive train is needed to pull those trains. Finally, in order to offer inexpensive piggy-backing opportunities for a train unit type, edges from previous iterations, with appropriate weights, must be included.

### 3.2 Minimum station turn around times

Station turn around times can be integrated into the flow model as follows: after having obtained a valid rostering from our approach, we check in the solution to see that minimum station turn around times have been obeyed. If they are violated, we artificially delay all incoming edges by the amount of time needed to carry out a coupling operation and thus create a new vertex; this has the effect that the train unit on this incoming edge then has a better chance at finding a next ride without violating minimum station turn around times. This yields a new graph, and we iterate this procedure until no coupling violations occur.

The number of iterations is only polynomial in this case as the minimum station turn around time is always less than four couplings. Thus, each vertex in the underlying graph will be delayed at most four times. In practice, the vertices at which violations occur are delayed by the maximum amount of time needed in the first iteration in order to save time.

We do not just add the delay to all stations, as at some stations, the delay is quite high ($\tilde{1}$ hour), and to do so would require extra train units in the rostering.

### 3.3 Additional costs

We can model fixed costs of each train unit type by simply making one unit of flow on the overnight-edges cost as much as the fixed costs for one train unit. Similarly, we can compute the costs for each each ride by computing and combining the costs per distance, time and ton kilometer for each train unit type. However, the costs that are incurred only for a train as whole (e.g. coupling/decoupling costs) and not for each train unit type are harder to model exactly. As an approximation, we allocate these train costs to a specific unit of the train, if possible the locomotive. This can lead to an extra cost in our calculations if two or more locomotives pull a train, but these are should be small compared to others.

## 4 Maintenance

Of the real-world constraints ignored by the standard minimum cost flow algorithm, maintenance is first and foremost. If an otherwise valid rostering solution ignores some costs, its real-world cost may increase, but the solution remains valid. The same does not hold for ignored constraints. Of constraints, maintenance is of topmost concern to the railway companies, as it cannot be solved locally (within one station), and it can greatly effect the cost and practicality of scheduling trains.

When considering maintenance, the straight-forward two-phase approach seems to break down. While many of the additional requirements can be accounted for by making modifications to the underlying graph in the network flow phase (as discussed in Section 3), the crucial maintenance requirements seem evasive to integration into the two-phase approach. We thus introduce a third phase into our approach that we call maintenance insertion. There are several approaches possible as to how to integrate maintenance in this third phase: some iterate over all three phases of our solution approach, others simply add a single phase after the two other phases have been completed. We propose three different approaches in Sections 4.1, 4.2, and 4.3. The complexity of the problem instance will determine which approach is most applicable.

### 4.1 Locally added maintenance

If the underlying network structure is rather simple, i.e., consists of only a few lines with abundant maintenance stations, we can hope for the best: the rotations obtained from the standard two-phase approach may already fulfill all or most maintenance requirements as the train units sometimes remain at maintenance stations for long enough time periods to perform required maintenance. In this case, a promising approach for integrating maintenance is to iteratively add "maintenance edges" into our network flow graph. Using this approach, we can iteratively run the two-phase solution on the new modified underlying graph until all needed maintenance is performed.

This "local fix" approach iterates the following basic steps:

1. Solve train length problem using network flow.
2. Make a train assignment to the train length problem.
3. Test the solution for maintenance requirement violations.
4. Introduce new maintenance edges into the graph.

Once the solution no longer violates any maintenance requirements, the algorithm stops. All steps except for the introduction of maintenance edges work exactly as in the basic approach described earlier. We introduce new maintenance edges into the graph as follows: for each rotation $R$, consider some point in $R$ when a maintenance constraint is first violated. For $k$ previous routes in the rotation ($k$ can vary), check to see if the train was in a station which could have performed the the maintenance type which is violated. If so, schedule this maintenance.

If maintenance cannot be scheduled so easily, the underlying graph is modified. Consider the last valid station before the violation such that the train still would have been allowed to travel the distance to the closest maintenance station. From this station to the maintenance station, we add a false route into our underlying time-schedule, which will be a "maintenance edge" in our network graph model. If the edge gives enough time to get to the maintenance station, and then to perform maintenance, and then another edge leads from the end of this maintenance back into the rotation we were considering, we can model this maintenance by the extra edges. We can encourage this maintenance either by giving these edges negative weights, or force it by setting the required flow over such edges to one.

This local fix approach seems to work quite well for problem instances in which maintenance really is not the central issue, with simple to follow maintenance constraints In these cases, minor detours will allow all needed maintenance. Unfortunately, maintenance requirements are rarely so simple.

### 4.2 Extra trains for maintenance

In this approach, we try to model a strategy of the railways: always try to keep a fully serviced train unit (of any type needed) available at each maintenance station. These trains will be rotated into rotations as needed to eliminate maintenance violations. The largest goal is to use as few spare trains as possible. This approach is non-iterative and consists solving the train length and assignment problems without maintenance, and then by adding extra trains into the assignment to alleviate maintenance requirements.

To add these extra trains, once again consider the last route completed by a train before the violation occurs. At this station, swap the train which needs maintenance for one which has just been maintained, and continue the same rotation with the new train. The train which needs maintenance is then routed to a maintenance station, maintained, and will later be swapped into another (or the same) rotation when another maintained train is needed.

In this approach, we keep the original cycles intact, which mimics the strategy of DB and SBB. It can be thought of as solving the rostering problem with virtual trains, needing no maintenance. The number of extra trains needed to maintain this illusion increases with the difficulty of the maintenance constraints. While conceptually simple, examination of implementation details will shows hidden complexities, especially concerning how to best link the start and end of the rotations. Simply linking the first and last stations in a cycle would not allow the cycle to "start" the cycle with a maintained train, and some care must be taken to keep feasible solutions while not simply requiring an extra maintenance.

### 4.3 Iteratively fix rotations with maintenance

Our last approach to maintenance combines some aspects of both preceding strategies. As in Section 4.1, we try to fix violations by adding maintenance into cycles for free when possible, and by routing trains to maintenance stations when it is not. Once maintenance is completed on the train, we move this train back into the same cycle, allowing the cycle to continue later on with a freshly maintained train as in Section 4.2. This altered cycle will not service every route on the original cycle, but the routes it does cover will be serviced with maintained trains. These routes are removed from the schedule, and solutions for the remaining routes, excluded from the altered cycles, will be iteratively found by starting over. Of the three approaches, this worked best overall for our test data.

## 5 Experiments

Here we discuss our test data and experiments. The major goal of these experiments was to carefully revise our model, test its practicality, and to ensure that important details were not being glazed over as they had been in previous, purely theoretical, treatments.

The overall goal in each experiment is the same: the minimization of the total costs. However, some of the test data given to us by the railway companies was chosen to test the minimization of train units, while other test sets were designed to test whether or not our maintenance strategies give feasible solutions.

Although the overall goal is the minimization of the total costs, our results summarized in Table 1 do not show these cost values. That is simply because the railway companies do not disclose the costs of their currently running solutions, and without the real values a comparison is not possible. At least, we compare the number of train units, which is the most important cost factor.

### 5.1 Test sets

We used five different test sets in our experiments, four of them a friendly contribution from DB and one from SBB. To simplify further discussions on these test sets we enumerate them:

- *BR112:* This set contains only locomotives of the same train unit type 112. A train unit in this test set is a single locomotive. The given schedule is a subset of the DB timetable with 2098 routes (each specified only by its end stations). While this test set is quite large, the maintenance requirements are not overwhelming, and the main objective is to minimize the number of train unites.
- *BR612:* This test set only contains diesel railcars (self-locomotive cars) of the same train unit type 612. Thus, the train unit consists of just one railcar. The given schedule is a subset of the regional timetable Saarland-Westpfalz in Germany with 332 routes. The difficulty here lies in the frequent refueling requirements of these locomotives, as there are only a small number of fueling stations, and they have limited capacity.
- *BR411:* This test set consists of two different Intercity Express (ICE) train units. The train unit type 411 is a composition of seven cars including a locomotive. The train unit type 415 is a similar but shorter composition. Two train units of the same type can be coupled together, but a substitution of a train unit of one type by the other is rarely possible. Both train types use the same maintenance infrastructures with capacity and time limitations. Testing maintenance feasibility is a main goal. The schedule is a subset of the international timetable between Germany and its neighbor countries with 496 rides. 330 of these rides must be conducted with train units of type 411.
- *BR218:* Our last DB test consists of all locomotives of the train unit type 218 and their associated passenger cars in the region Schleswig-Holstein. The given schedule contains 7666 routes of up to 32 different train unit types. Some of these 32 train unit types are only used as possible substitutions in case of car capacity problems. Here, we must schedule both train cars and locomotives, adding to the complexity of the problem. Additionally, the number of some train unit types are limited, and we must find appropriate substitutions in case of car capacity problems.
- *BR1210:* Our last test set consists of all self-locomotive train compositions of the Zurich S-Bahn. A train unit is in this test is a composition of several cars with locomotive. S-Bahn trains are either one or two coupled push-pull train units. The given subset of the SBB schedule contains 6151 routes. This test set is the only one which supplies information about the type of turn within a station. Because all train units are composed the same way, the additional information allows the computation of the first class car positions within each station. Besides minimizing the number of train units used, the position of the first class cars within the station should be the same from day to day on any route. Maintenance information was not supplied.

### 5.2 Results

The most interesting results of these test scenarios are the rolling stock rosters, but they are much to large to present here. We show a cutout of one of the rosters, and some values to summarize how efficient our results were.

Our software is built of 20000 lines of C++ code. Beside of standard libraries we only used LEDA (Library of Efficient Data types and Algorithms)[2]. We have run our tests on a PC laptop with a 1.6 GHz Mobile Pentium 4 processor, 512 MByte of RAM under Windows XP.

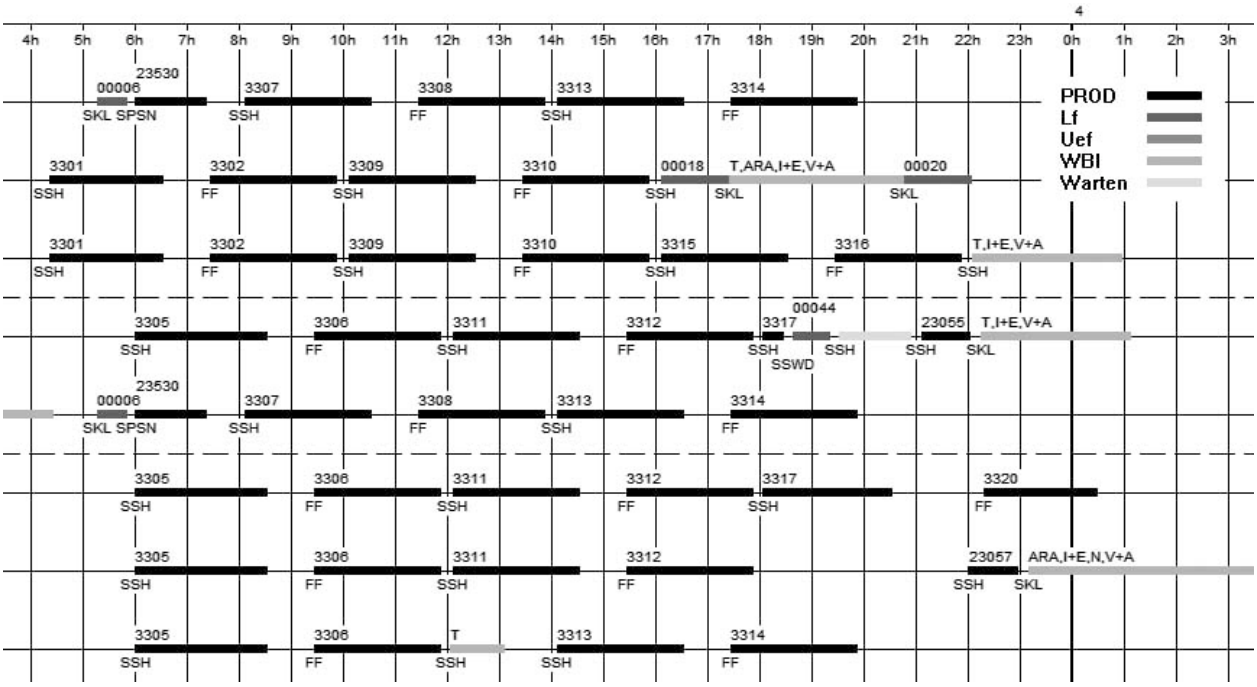


**Fig. 1.** Graphical representation of a rolling stock roster.

In Figure 1 you see a typical cutout of a graphical representation of a rolling stock roster. Several different depictions are meaningful, this format matches the one used by SBB. On the x-axis we see the time between 4:00 of the third day in period until 3:00 of the next day. Each line represents one train unit over one period and the bars on that line show the planned rides of the train unit. On the second line, for example, we see four consecutive productive rides starting and ending in SSH followed by an empty movement to SKL where the train unit is maintained. Several maintenance operations (T, ARA, I+E, V+A) are planned between 17:30 and 20:45. Finally, the train unit makes an empty movement to the next station where it is needed. Above the bars we see train identification numbers, labeling which train units belong to the same train. For example, the train 3305 starts at 6:00 in SSH and ends at 8:30 in FF. This train consists of four separate train units.

The dashed horizontal lines separate train units of the same rotation. For example, the first three train units are in the same rotation of length three.

In Table 1 we summarize our results of the five test sets. As would be expected, ignoring maintenance constraints allows the automatic construction of a solution at least as good as the ones in use. Unfortunately, our results with maintenance are not as good. In the two test sets with the most maintenance constraints, BR612 and BR411, our results with maintenance suffer the most, (33%) worse than the current solution of the railway companies. Especially in the case of BR612, this is not surprising: refueling is needed so frequently that it must be incorporated very efficiently to the schedule. Pulling a train out of rotation and traveling 50 km extra for each 10,000 km maintenance is very different than doing it for a 500 km tank of fuel.

[2] LEDA has been developed at the Max-Planck-Insitut für Informatik, Saarbrücken (http://www.mpi-sb.mpg.de/LEDA/) and is available at http://www.algorithmic-solutions.com/.

| Test set | Graph: $\|V\|$, $\|E\|$ | Number of train units in real solution | our solution without maintenance | our solution with maintenance | Runtime in seconds |
|---|---|---|---|---|---|
| BR112 | 12521, 90050 | 66 | 52 | 55 | 35 |
| BR612 | 1025, 165266 | 9 | 9 | 12 | 14 |
| BR411 | 411: 3602, 1066810 | 28 | 27 | 36 | 354 |
| | 415: 1124, 214768 | 8 | 8 | 11 | 67 |
| | Total | 36 | 35 | 47 | 421 |
| BR218 | C1: 3005, 11982 | | 6 | 7 | 6 |
| | C2: 3655, 14075 | | 9 | 10 | 29 |
| | C3: 6891, 40459 | | 10 | 10 | 10 |
| | C4: 304, 623 | | 1 | 1 | 0 |
| | C5: 6273, 22383 | 68 | 6 | 8 | 8 |
| | C6: 3013, 12703 | | 5 | 6 | 6 |
| | C7: 7048, 30324 | | 11 | 12 | 8 |
| | C8: 574, 1351 | | 2 | 3 | 2 |
| | C9: 8508, 62435 | | 16 | 19 | 27 |
| | 218: 22628, 368991 | 72 | 57 | 84 | 81 |
| | Total | 140 | 123 | 160 | 177 |
| BR1210 | 36895, 319211 | 120 | 79 | | 79 |

**Table 1.** Table of results.

## 6 Conclusion

We have presented an extended and therefore much more practical version of the standard fleet size problem. We have shown that many extenstions of the rolling stock rostering problem can be well integrated into the standard flow model approach. For a constraint which we cannot easily integrate (maintenance), we have developed some heuristics to modify the standard flow approach.

We have implemented our three phase approach and tested it with real data from the German Railway and Swiss Federal Railway. Although our solutions are worse than the rostering solutions already used by those railways, our experiments are not failures. It would be overly optimistic to assume that the first attempt of a fully automated system would improve upon long tested and used solutions. Each year, several (or many) man years of labor go into refining rostering solutions for minor scheduling changes made year to year. Over the years, it may well be that minor scheduling changes have also been made to accommodate the rostering solutions. Although finding improved fully automated solutions is an ultimate goal, a more immediate goal was to build a system which could be used interactively, by schedulers, to help in their jobs. Given an automated solution, the scheduler can pull out partial solutions which they believe look promising, and then run the system iteratively on the parts which looked worse. Given our quick runtime, on modest equipment, this interactivity is quite feasible.

## References


[BHR99] P. Brucker, J.L. Hurink, and T. Rolfes. Routing of railway carriages: A case study. In *Memorandum No. 1498, Fac. of Mathematical Sciences*. University of Twente, 1999.

[DDSS95] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Time constrained routing and scheduling. In *Handbooks in OR & MS*, volume 8, pages 35–139. Elsevier, 1995.

[DF54] G. Dantzig and D. Fulkerson. Minimizing the number of tankers to meet a fixed schedule. *Naval Research Logistics Quarterly*, 1:217–222, 1954.

[EGH+01] T. Erlebach, M. Gantenbein, D. Hürlimann, G. Neyer, A. Pagourtzis, P. Penna, K. Schlude, K. Steinhöfel, D. Taylor, and P. Widmayer. On the complexity of train assignment problems. In *ISAAC: International Symposium on Algorithms and Computation, LNCS*. Springer-Verlag, 2001.

[Gan01] M. Gantenbein. The train length problem. Diploma Thesis, Department of Computer Science, ETH Zürich, 2001.

[HS89] S. Hochbaum and A. Segev. Analysis of a flow problem with fixed charges. *Networks*, 19:291–312, 1989.

[Sch93] A. Schrijver. Minimum circulation of railway stock. *CWI Quarterly*, 6(3):205–217, 1993.