# Railway Crew Pairing Optimization

Lennart Bengtsson, Rastislav Galia, Tomas Gustafsson,
Curt Hjorring, and Niklas Kohl[⋆]

Jeppesen AB, Odinsgatan 9, Göteborg, Sweden
curt.hjorring@jeppesen.com

**Abstract.** The use of automatic crew planning tools within the railway industry is now becoming wide-spread, thanks to new algorithm development and faster computers. An example is the large European railway Deutsche Bahn, which is using a commercial crew planning system developed by Jeppesen (formerly Carmen Systems). This paper focuses on the crew pairing problem that arises at major railways. Even though it is similar to the well-studied airline crew pairing problem, the size and complexity of the railway operation necessitates tailored optimization techniques. We show that a column generation approach to the pairing problem, which combines resource constraints, $k$-shortest path enumeration and label merging techniques, is able to heuristically solve a 7,000 leg pairing problem in less than a day.

## 1 Introduction

The process of crew planning at large transportation companies, such as railway and airline companies, is often very complex. Feasible work schedules have to comply with a large set of company rules and union agreements, and might also take into account preferences of individual crew members. The schedules should also take into consideration the available number of crew in each crew depot.

In order to manage the complexity and reduce the crew costs, some railways and most airlines have started to use automated crew planning tools. The Carmen crew scheduling system is successfully implemented and used by many of the world's largest transportation companies. Railway customers include the German state railways Deutsche Bahn, the Swedish State Railways and the freight operator Green Cargo. In the airline sector, the system is used by all major European airlines, as well as several operators in North America and Asia.

Compared to manual planning, automated planning tools have a number of advantages. As already mentioned, sophisticated optimization techniques can be used to reduce the crew costs. It is also possible for the planning department to produce crew schedules for several scenarios in parallel, and then pick the most suitable one for production. The ability to evaluate "what-if" scenarios can be used for strategic planning and timetable changes etc.

A challenge when going from manual planning to automatic, is to model the many rules which are soft in the sense that "they should not be broken unless it's

[⋆] Now at DSB Planning, Sølvgade 40, DK-1349 Copenhagen K, Denmark.

necessary". Especially for railways, in manual planning there are frequently cases where the planner violates rules based on experience and "common sense". In an automated system, this means that we have to model and express the many "common sense" trade-offs with penalties, and end up with a highly complex cost-function for the optimizer.

A trend in the industry is a drive for incorporating timetable changes very late in the planning process. In addition, one would like to be able to better adjust the number of conductors on a train to the daily or weekly passenger demand. Altogether, this means that the turn-around time and flexibility of the automatic tool is very important and the performance of the optimizer is essential.

In the following we will focus on the railway crew pairing problem. The techniques described have been successfully applied to planning problems from several rail operators. We exemplify our approach with data from Deutsche Bahn (DB), which in our experience has been the most challenging with respect to both size and modeling complexity. DB, which is one of the world's largest transportation companies, consists of a number of partly independent companies which operate regional and commuter traffic, together with the long-distance operator DB Fernverkehr. The total number of crew (train drivers and conductors) is around 30,000, distributed across more than 100 crew bases. A train is operated by one train driver and from zero to seven conductors depending on the type of train, the expected number of passengers, and a number of other factors.

## 1.1 The Railway Planning Process

The operation of a passenger railway is defined by the timetable, the allocation of infrastructure such as tracks and platforms to the timetable and the allocation of rolling stock to the timetable. Timetable, infrastructure and rolling stock are typically planned well in advance before the actual operation and often a plan is valid for an entire timetable period of six months or more. However, modifications, often due to maintenance work or special events, must occasionally be introduced in to the plan.

The timetable and the rolling stock schedule together define the basic crew need to operate the trains. Basic crew need includes the locomotive driver and, for passenger trains, the minimum number of conductors required. In addition to the basic crew need, extra crew might be required, depending on factors such as the type and quantity of rolling stock, the type of service offered on the train and the expected number of passengers. These factors also determine the qualifications required by the crew members assigned to the train.

Because a train might depart early in the morning and arrive at its destination late in the evening, it is necessary to divide the trains into *legs* (pieces of work the cannot be divided). Legs start and end at stations where it is possible to change crew. The goal of railway scheduling is to assign the legs to qualified named individuals such that rules and regulations with respect to working time, rest time etc. are satisfied, and such that costs and other quality aspects of the solution are optimized. The problem naturally decomposes into the crew pairing problem, in which pairings, i.e. sequences of legs, starting and ending at a crew base are constructed, and the crew rostering problem, where the anonymous pairings are assigned to named individuals. Furthermore, the pairing problem can be decomposed into a daily problem, in which pairings that cover all legs that are operated more or less on a daily basis are constructed, and a roll-out process, in which the daily solution is "rolled out" over a larger planning period, followed by a second planning phase that takes care of any irregularities in the timetable.

On the level of detail introduced so far, the airline and the railway crew scheduling problems are very similar. This is especially true for the rostering problems, which mainly have to deal with rostering rules and crew qualifications, and are less exposed to the underlying transportation network. A more detailed comparison of the pairing problems reveals a lot of similarities, but also important differences.

### 1.2 Outline of This Paper

After briefly reviewing existing literature on crew scheduling problems in Sect. 2, we present a mathematical model of the general crew pairing problem in Sect. 3. This model applies to both airline and railway problems. In Sect. 4 we describe how the general pairing problem can be solved with a column generation approach. A more detailed description of a typical *railway* crew pairing problem can be found in Sect. 5. In order to successfully solve railway pairing problems, Jeppesen has introduced some modifications in the general column generation scheme; these are presented in Sect. 6 along with some illustrative results. Then we round off the paper with large-scale railway planning results in Sect. 7 and some conclusions in Sect. 8.

## 2 Previous Work — Literature Review

Among the crew scheduling problems within the transportation industry, the airline crew pairing problem has got a lot of attention in the operations research literature. Finding an optimal set of pairings can be formulated as an integer programming problem, in which each column represents a feasible pairing. Unfortunately, the number of columns is immense. Snowdon et al. ([12]) report that a daily problem for a large American carrier has roughly $10^{14}$ columns. Therefore, delayed column generation is frequently used. An early application of this is due to Minoux ([11]), in which it is shown that columns can be priced by solving a shortest path problem on a network with arcs representing flights and overnight connections. A limitation of the flight network approach is that the network only captures flight connection rules, so rules that affect working days as a whole, or the entire pairing, are ignored. Therefore only a small fraction of the paths in the flight network form legal pairings. In the airline business, the sequence of flights flown in a typical working day is known as a *duty period*, and for many rules the legality of a single duty period is independent of preceding or succeeding duties.

Lavoie et al. ([9]) take advantage of the structure and form a duty period network where nodes are duty periods with state information, and arcs represent legal overnights. A similar approach was later used by Desaulniers et al. ([5]) to solve crew pairing problems at Air France. Vance et al. ([13]) present results for both flight and duty based implementations. Resources are added to each node in the networks to track additional legality conditions, and *resource constrained shortest path* problems are solved. Comparisons between the two versions are difficult to make since they implement different variants of the rules, but in general the flight based version spends a larger portion of time in the pricing routine than does the duty based version. However, the duty version cannot solve as large problems as the flight version due to memory limitations. Storage of the duty periods and the duty period connections is prohibitive for large problems (there is one arc for each legal duty period connection).

The excessive memory usage of the duty network is addressed in Hjorring and Hansen ([7]) by creating an initially relaxed network where duty-duty connections are replaced by flight-flight connections. Portions of the network are dynamically refined when required, in order to capture additional cost and legality information. The method also uses a $k$-shortest path approach, instead of resource constraints. This allows rules and costs to be implemented in a separate module that presents a black box interface to the pricing subproblem. The rules module can then be implemented using a modeling language that allows the user to easily write their own rules and cost function.

Until recently, the existing literature on railway optimization problems was mainly focused on vehicle scheduling problems, see for example the review by Cordeau et al. ([4]). The large size and the complexity of the crew scheduling problems that arise in the railway business made it difficult to apply optimization methods for crew planning. This has now changed, thanks to new algorithm development and faster computers. An example is the project "Destination: Customer" at the Dutch railway operator NS Reizigers, which aims at increasing the quality and punctuality of passenger trains, see Kroon and Fischetti ([8]). The authors present two approaches for solving the crew pairing problem: a greedy approach based on constraint programming, and a column generation approach based on reduced cost generation of duties combined with a subgradient optimizer and a variable fixing scheme. Another example is the development of a combined crew pairing and rostering system for the Italian state railways, see Caprara et al. ([2,3]). The scheduling process is divided into three phases: enumeration of all feasible pairings, solution of a large set covering problem using Lagrangian heuristics and variable fixing, and the production of cyclic *rosters* (sequences of pairings and weekly rest periods) which are subsequently distributed among the crew members.

Wren et al. ([15]) focuses on the problem of scheduling drivers for short-distance trains and buses. They stay away from the column generation approach, because they believe that the cost function is not sufficiently decomposable. Instead they enumerate feasible duty days explicitly, and apply filtering heuristics to limit the number of duties that need to be sent to the optimizer. The planning system, called TRACS II, appears to be in use at a number of British bus and railway companies.

## 3 General Problem Definition and Terminology

In this section we will introduce some terminology and give a mathematical formulation of the general crew pairing problem.

Let $\mathcal{L}$ denote the set of (leg, function)-combinations. A leg may require several crew members, but each will be assigned in a unique function, so for each function-leg $l$, $l \in \mathcal{L}$, exactly one crew member should be assigned. In the literature function-legs are sometimes called “trips” or “increments of work”. $\mathcal{P}$ denotes the set of legal pairings. A pairing $p$, $p \in \mathcal{P}$, is an ordered set of function-legs, which can legally be operated by a crew member. A pairing will also contain other activities, such as preparation and closing activities, and may contain so called deadheads. A deadhead is a passive transport, either on a leg on which other crew work, or by some other means of transportation. A pairing must start and end at the same crew base and must comply with all rules and regulations regarding work time, rest, qualifications, etc. The coefficient $a_{lp}$ is 1 if pairing $p$ contains function-leg $l$ and 0 otherwise. The cost of a pairing is denoted by $c_p$ and may be the sum of real costs associated with the operation of the pairing and penalties capturing robustness and social aspects.

The main constraint of the crew pairing problem requires all function-legs to be covered by exactly one pairing, but the solution space will also be limited by other constraints. For this purpose we introduce the set $\mathcal{H}$ of hard constraints and the set $\mathcal{S}$ of soft constraints. The contribution of pairing $p$ towards a hard constraint $h$, $h \in \mathcal{H}$, and a soft constraint $s$, $s \in \mathcal{S}$, is denoted $b_{hp}$ and $b_{sp}$ respectively. The sum of constraint contributions over pairings in a solution is bounded by $d_h$ and $d_s$ for hard and soft constraints respectively. A soft constraint $s$ can be violated at the cost of $c_s$. These types of constraints are often used to ensure a distribution of the pairings corresponding to the distribution of crew and to ensure an acceptable portion of pairings with particular unpopular properties such as night stops.

The decision variables of the model are $x_p$, which is 1 if pairing $p$ is used in the solution and 0 otherwise, and $y_s$ which determine the violation of soft constraint $s$. The problem can now be stated as

$$\min \sum_{p \in \mathcal{P}} c_p x_p + \sum_{s \in \mathcal{S}} c_s y_s, \quad \text{s.t.} \tag{1}$$

$$\sum_{p \in \mathcal{P}} a_{lp} x_p = 1, \ \forall l \in \mathcal{L} \tag{2}$$

$$\sum_{p \in \mathcal{P}} b_{hp} x_p \leq d_h, \forall h \in \mathcal{H} \tag{3}$$

$$\sum_{p \in \mathcal{P}} b_{sp} x_p - y_s \leq d_s, \forall s \in \mathcal{S} \tag{4}$$

$$x_p \in \{0, 1\}, \ y_s \geq 0 \tag{5}$$

The objective function (1) specifies that we want to minimize the sum of the cost of selected pairings and the cost of violating soft constraints. Constraint set (2) requires each function-leg to be covered by exactly one selected pairing. Constraint set (3) requires all hard constraints to be satisfied and constraint set (4) ensures that $y_s$ is at least equal to the violation, if any, of the soft constraint $s$. Constraint set (5) requires $x$ variables to be binary and $y$ variables to be non-negative. The model is a generalization of the well known set partitioning problem.

The model presented above is not exactly identical to the one solved by the Carmen Crew Pairing system. The modifications can be summarized as follows:

- In constraint set (2) equality is replaced by greater than equal, i.e. with covering constraints. This is a relaxation, but typically a covering solution, i.e. a solution where the left hand side is more than 1 for at least one $l$, can be transformed to a partitioning solution, by substituting a function-leg with a deadhead on the same leg. Generally, the less constrained covering formulation is easier to solve. In cases when this relaxation is not valid, for instance when a legal deadhead cannot be found, the penalty for overcovering the active leg can be gradually increased.
- All constraints are soft. For constraint sets (2) and (3) a very large penalty for violation is defined. This is because a solution that violates a hard constraint is much more useful from a practical point of view than no solution at all. Typically a solution with hard constraint violations will hint at problems with the input data.

## 4 Algorithms for the General Pairing Problem

The problem discussed in Sect. 3 is generic and can be applied to both airline and railway crew pairing problems. In this section we outline how Carmen Crew Pairing solves the general problem.

The set covering problem in Sect. 3 is difficult to solve in practice, for two reasons. To start with, the number of columns in the constraint matrix $A$ tends to be huge, so it is not feasible to enumerate all of them. One either has to limit the enumeration to some cleverly chosen subset of the feasible pairings, or generate columns "on demand" using optimizer feedback. In addition, large set covering problems are difficult to solve with standard integer solvers, so one has to resort to heuristic techniques.

Carmen's approach to the pairing problem is to generate pairings dynamically using reduced cost feedback from the set covering optimizer, a technique sometimes referred to as Dantzig-Wolfe decomposition in the literature. The goal of the decomposition is to separate the "complicating" problem constraints (the pairing rules) from the "simple" constraints (2)– (5) above. The "simple" constraints form the *master* optimization problem, whereas the "complicating" constraints are hidden inside a column generation subproblem, usually called the *pricing* problem. In this context, the set covering problem in Sect. 3 and the master problem is the same.

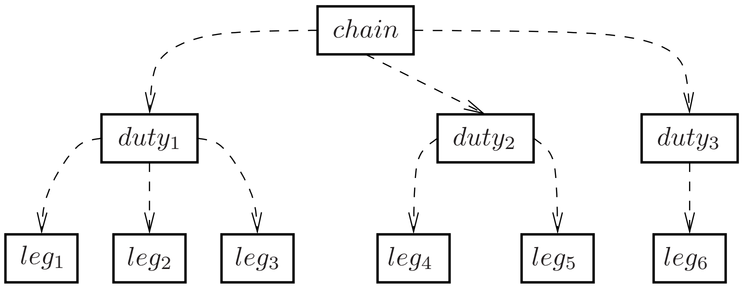


**Fig. 1.** Hierarchy of chain levels

## 4.1 The Pricing Problem

The goal of the pricing problem is to find legal pairings with negative *reduced cost*, for addition to the master problem. The reduced cost includes feedback from the optimizer (the pairing cost is reduced by the sum of the dual variables of the constraints that the pairing covers.) Because the pricing problem is intimately connected with the pairing rules and the cost function, we start with describing the general structure of the pairing rules and the pairing cost function.

The Carmen Crew Pairing system (CCP) contains a rule modelling language (Rave) that permits planners to easily add and modify the rules and the cost function. Rave is implemented as a black-box system, which is able to decide whether a pairing is legal with respect to the rules, and to evaluate the pairing cost. However, it does not expose the internal details of the evaluation process to the rest of the system.

For modelling purposes, a sequence of legs is usually grouped into a hierarchy of levels, as shown in Fig. 1. In the pairing problem there is usually at least one intermediate level of *duties* between the atomic level of the legs and the top–most pairing level. The levels are used when defining rules and costs: certain rules are leg dependent and need to be evaluated for each leg in the chain, whereas other rules may be duty- or chain-dependent.

A related concept is the range of objects that evaluated expressions depend on. For example, a rule that checks the min connection time between two legs has very short range: only two legs. In the other extreme, an expression counting the length of the pairing in days has chain range.

The column generator that solves the pricing problem relies on $k$-shortest-path enumeration within a network, in which the arcs are of two types: block arcs, which represent partial pairings, and connection arcs, which connect partial pairings together, see Fig. 2 for an example. The block is defined by an intermediate level in the Rave language. Preferably, the majority of the paths in the network should correspond to legal pairings; otherwise the efficiency of the column generator is reduced. Therefore the block level needs to be selected carefully. The most constraining pairing rules should have a range of at most one or two blocks; they will then be captured accurately by the network. Often, a suitable block level is equal to one day of work (a duty), but there are other possibilities.

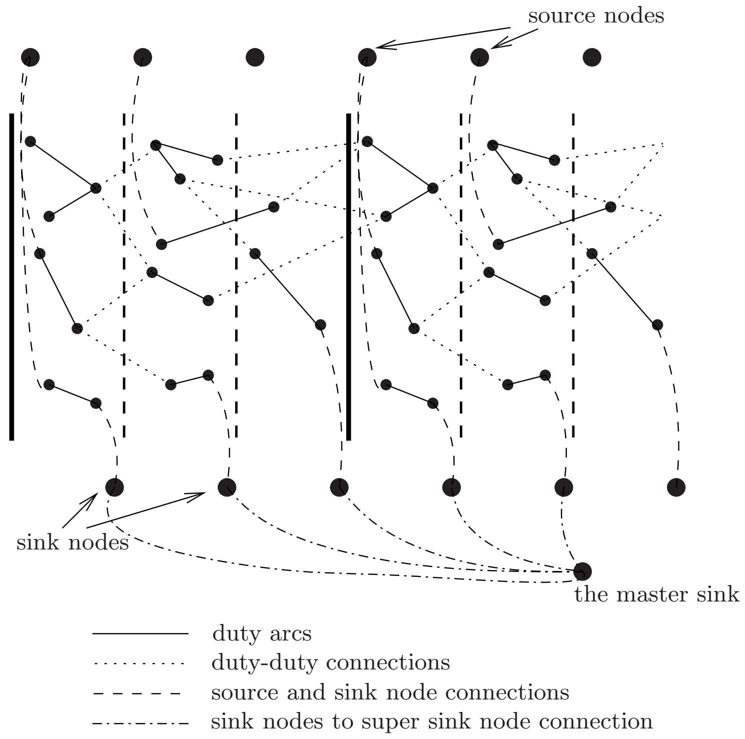

source nodes
sink nodes
the master sink
duty arcs
duty-duty connections
source and sink node connections
sink nodes to super sink node connection

**Fig. 2.** Schematic view of the pricing network. There is a pair of source and sink nodes for each calendar day and homebase combination. For a given start day, the master sink node connects all sink nodes that represent legal ending days, assuming that there is a max calendar length rule in the ruleset.

In a similar fashion, we assume that the majority of the cost terms in the cost function can be decomposed into block and block-block contributions, which can be stored on the arcs in the network.

In one iteration of the column generator, paths with negative reduced costs are enumerated in increasing cost order, using a $k$-SP algorithm. For each path found, the Rave system evaluates the true pairing cost, which might differ from the network estimate, and verifies that the pairing is legal. If the pairing still looks good, it is added to the master problem. The process continues until no more attractive (negative cost) paths can be found, or if “sufficiently” many pairings have already been added. Because there are multiple source nodes in the network (c.f. Fig. 2), each base and starting day combination is treated separately.

## 4.2 The Master Problem

After each call to the pricing routine, the newly generated columns need to be incorporated into the master set covering problem defined in Sect. 3. The solution of the LP-relaxed master problem yields a set of dual variables that are sent to the pricing problem in the next iteration. We iterate over pricing calls and solutions of the master problem until no more attractive columns can be found. At that point, we have arrived at the optimal LP solution of the pairing problem. Typically, the LP solution is fractional, and only yields a lower bound of the pairing solution cost. Branching schemes or integer heuristics can be used to find good integer solutions, given the set of columns found so far.

Small set covering problems may be solved successfully by commercial IP solvers, such as CPLEX or XPRESS, but they tend to become unreasonably slow for problems of realistic size. The Carmen pairing optimizer relies entirely on specialized solvers. They are described in more detail in Sections 6.6 and 6.7.

## 5 The Railway Crew Pairing Problem

After the presentation of the general crew pairing problem in the previous sections, we now consider the *railway* pairing problem in more detail. This section focuses on the Deutsche Bahn planning problem, which is an example of the large and complex planning problems that arises at major railways.

### 5.1 Modelling of Rules, Regulations and Objectives

It is not possible to give a complete account of all rules, regulations and objectives of the DB crew pairing planning within the context of this paper. The purpose of this subsection is just to outline some of the more important rules, regulations and objectives. It is also the case that the labour agreements change from year to year, putting very high demands on the flexibility of the system.

In addition to function-legs and deadheads, a pairing will contain a number of derived preparation and closing activities. The derived activities can be calculated from the function-legs and deadheads and the rolling stock rotations.

The construction of individual pairings is mainly limited by a number of work, rest and connection time rules. There are three main different time concepts. There are activities associated with starting and ending a duty, starting and ending a train and walking times, for example between trains and the rest facilities. In addition to these times there is a required connection (buffer) time when crew change from one train to an other.

The *duty time* is the time from duty start to duty end. *Work time* is the time when the crew member carry out activities that according to German legislation are classified as work. *Paid time* is the time for which the crew member is paid. All work time is paid, but some non-work activities, primarily deadheading and paid rest, are paid without classifying as work time. Work time is generally limited to 10 hours per duty-day. A duty with at least 6 hours of work time requires at least 30 minutes of rest time, and a duty with at least 9 hours of work time requires at least 45 minutes of rest time. Rest must generally be allotted at a station with rest facilities, but can also be planned on-board a train en-route. The possibility of using on-board rests is constrained by several factors, but especially complicating is the fact that only a limited number of crew can rest at the same time. Required rest is usually unpaid, whereas any additional rest while connecting between trains is paid.

A pairing may contain up to two duty days. Nightstops must last for at least 5 hours, and are only permitted at certain stations. If the nightstop is shorter than 9 hours, the total work time of the pairing is limited to 10 hours.

Not all train products can be operated from all crew bases and certain trains must be operated from a particular base. This is in particular the case for some of the international traffic. There are also limitations on to which extent different function-legs can be mixed in the same pairing. It could otherwise happen that the rostering problem becomes infeasible, because no crew members at all are qualified to work on certain pairings.

The objective function contains monetary terms including paid time, hotel costs and deadheading costs as well as a large number of terms capturing operational stability and crew preferences. These include

- A train should not change crew too often, so a working period of less than e.g. two hours on a train is penalized.
- For the same reason, changing crew near the start or end of a train is penalized.
- Pairings with a long nightstop relative to the total paid time on the pairing are penalized.
- Pairings where the duty after the nightstop is longer than the duty before the nightstop are penalized.
- Changes between different functions and different train products are penalized, in order to reduce the number of different qualifications that are needed to work on individual pairings in the pairing solutions.
- Legs that only require a team of two conductors should preferably not be mixed with legs that require three conductors, because then the benefits of teaming will be reduced, c.f. Sect. 5.4.
- There is a fixed cost per duty day. This is to decrease the total number of duty days and consequently increase the paid time per duty day. Pairings with much paid time per duty day are easier to roster.

In total the DB ruleset contains about 100 separate rules. All rule and cost definitions are expressed in the Jeppesen Rave modelling language. Currently, the DB Rave code consists of $\approx 30{,}000$ lines distributed over 50 modules. Thanks to the black box system, there is a clear separation between the optimizer core and the Rave code, a fact that greatly simplifies maintenance and allows the Rave code to be developed independently of the optimizer. Integrating the rules and costs directly into the optimizer would have been a formidable task, with little hope of commercial success.

## 5.2 Base Constraints

Clearly, a solution must consist of legal pairings, but there are additional constraints on the total solution. These are modelled with constraints of type (3) or (4), c.f. the mathematical model discussed in Sect. 3. The most important of these are base constraints and constraints on the number and distribution of nightstops. The purpose of the base constraints is to ensure a distribution of pairings, and consequently of the workload, between the bases, which corresponds to the actual distribution of crew. In the long and mid-term run the distribution of crew can be influenced by operational efficiency, but in the short run the crew distribution is fixed. For each crew base there is a maximum and possibly a minimum amount of paid time, which can be assigned. There might also be separate base constraints for crew with special qualifications. Pairings with nightstops are permitted and economically quite attractive, but are unpopular with crew. Therefore it is necessary to introduce limits on the total number of pairings with nightstops as well as on the distribution of these between crew bases and between crew groups with different qualifications.

### 5.3 Variable Crew Need

In practice, the number of crew members that should work on a leg is not always given a priori. For service reasons, an additional crew member may be desirable, but not required. If the number of expected passengers vary substantially between legs on the same train, the required crew may also go up and down, but for practical reasons, the staffing is kept at a constant level, if this is not costly. This could be modelled by partitioning the set $\mathcal{L}$ in Sect. 3 into a required set of function legs, for which equality in constraint set (2) must be satisfied, and a set of optional function-legs, where relatively cheap slack variables are introduced.

### 5.4 Teaming Aspects

From an operational point of view, it is desirable that crew members work in teams, because then it becomes more likely that all required crew members arrive at a particular train departure station in time. Teaming is also preferred for social reasons. It is therefore necessary to take teaming aspects into consideration when solving the pairing problem. A typical approach is the following. Firstly team master pairings that cover all trains that require teams are created. For each pairing in the master solution, the minimum supplementary crew need is calculated, and then a number of copies of the master pairing, one for each extra crew member in the team, are added to the pairing solution. Finally, pairings for supplementary crew are created to cover any remaining crew need.

## 6 Algorithmic Contributions for the Railway Pairing Problem

### 6.1 The Pricing Network of Trains

As we have seen in Sect. 4.1, the pricing problem is transformed into a shortest path problem within a network of partial pairings ("blocks"). A question that arises when designing a pricing network for railway pairings is what a "block" in the network should represent. In the airline case the duty day is the natural candidate. However, it turns out that the number of legal duties in a typical railway problem is enormous, so it is not feasible to create the full duty network. Instead we choose to let the blocks in the network represent work periods on the same train. Whereas this approach leads to a manageable network size, duty dependent rules are not captured accurately in the network. Consequently, a large fraction of the paths found by the $k$-SP routine will violate duty time or work time constraints. This, in turn, could severely impact the performance of the column generator and the solution quality. In the following paragraphs we discuss how the performance degradation can be avoided.

### 6.2 Resource Dependent Cost Elements

The resource constrained shortest path problem has traditionally been used as a subproblem in applications of column generation to scheduling problems. The technique can be extended to model some classes of nonadditive costs as well.

Let us assume that the cost of a pairing can be written in the form $c(p) = c_a(p) + c_n(r(p))$, where $c_a(p)$ captures the additive parts of the cost function, and is additive with respect to block and block-block connections. Here $c_n$ is a, possibly non-linear, function of the resource vector $r(p)$.

Not every cost function can be modelled using resources. The main limitation is that the resource vector $r(p)$ must itself be additive, because it needs to be modelled accurately by the network. Furthermore, the non-additive part of the cost $c_n$ is required to be a non-decreasing function with respect the the individual resource components. The last *convexity* requirement enables certain short-cuts in the $k$-SP enumeration, as described below.

The current network topology also allows $c_n$ to depend on some other characteristics of the pairings, such as the starting day of the pairing (determined by the source node of the path) and the length in calendar days. Some examples of costs modelled by resources:

- Penalize pairings for which the work time exceeds a limit by a constant value.
- If the number of short night stops in the path exceeds one, and the total work time of the pairing is above 10:00, add a very high penalty to the cost of the pairing.

The latter example illustrates how the rule concerning max work time in pairings with short night stops (c.f. Sect. 5.1) can be modelled using a pair of resources.

### 6.3 Extension of the $k$–SP Algorithm

The pricing network, as described in Sect. 4.1, needs to be modified in order to handle a resource-dependent non-additive cost function. Instead of storing the total reduced cost on the arcs, we store the additive part $\bar{c}_a$ of the reduced cost, together with the resource vector $r(p)$.

The pricing routine enumerates paths in *additive cost* order. For each path found, the resource vector $r(p)$ is summed up along the path and the non-additive part of the path cost is evaluated using a Rave definition. Only if the total path cost is attractive, will the cost and legality be evaluated by Rave.

Because the $k$-SP routine only "knows" about the additive part of the cost, a potentially large fraction of the enumerated paths might turn out to be non-attractive when the non-additive part has been added. We use the concept of *non-dominated* paths to significantly reduce the number of useless paths. Consider the graph in Fig. 3. Two sub-paths can be found from the source node $n_s$ to the intermediate node $n_i$, with (cost, resource) sums $(-2, 50)$ and $(2, 60)$, respectively. Because the second sub-path has both higher cost and higher resource consumption than the first sub-path, we can conclude that the *total* cost of the second path, extended to the sink node $n_d$, is going to be higher than the total cost of the first path. Here we have to rely on the assumption that the non-additive function is non-decreasing. We say that the second path is *dominated* by the first path.

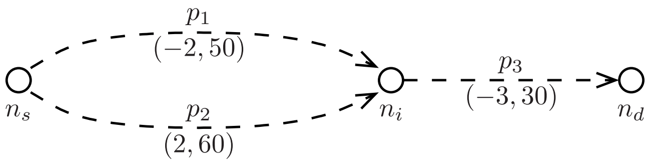


**Fig. 3.** Impact of the resources on the $k$–SP algorithm

More generally:

**Definition 1.** *The path $p$ is said to be* dominated *by path $p'$, if $\bar{c}_a(p) \geq \bar{c}_a(p')$ and $r_j(p) \geq r_j(p')$ for every resource $r_j$.*

**Definition 2.** *Let $P(n_s, n)$ be the set of all paths from the source node to node $n$. A path is called* Pareto-optimal*, if it is not dominated by any path in $P(n_s, n)$.*

We modify the $k$-SP routine so that it skips the dominated paths in the network. The dominance tests are done at all nodes in the network, and take advantage of the fact that paths are generated in increasing cost order. A variant of the algorithm has been described by Azevedo and Martins ([1]).

If the set $P(n_s, n_d)$ of paths from the source node $n_s$ to the sink node $n_d$ contains at least one attractive path, then it also contains at least one attractive Pareto-optimal path. This means that the pricing subproblem is guaranteed to find at least one attractive path, if any at all exist.

## 6.4 Label Merging

Because the number of arcs in the pricing network can exceed 1,000,000 for a typical daily DB pairing problem, the number of non-dominated path labels created by the $k$-SP routine might exceed memory limitations. A simple adaptive technique can be used to regulate the number of generated non-dominated paths in the network. The underlying idea is that we want to "merge" resource vectors in the Pareto-optimal set that are almost identical. To be more specific, path $p$ is said to be dominated by path $p'$ if $\bar{c}_a(p)+\kappa_a(n_s) \geq \bar{c}_a(p')$ and $r^{(j)}(p)+\kappa^{(j)}(n_s) \geq r^{(j)}(p')$ for each resource $r^{(j)}$, where $\kappa_a(n_s), \kappa^{(j)}(n_s)$ are positive parameters depending on the current source node $n_s$. The parameters $\kappa_a(n_s)$ and $\kappa^{(j)}(n_s)$ are adaptively changed after every pricing iteration, depending on the time spent in a pricing routine for source node $n_s$.

## 6.5 Impact of Improvements

The extensions to the network model and pricing subproblem solver that were mentioned in Sect. 6 have been tested on a number of problem instances, and we observe significant performance improvements. Fig. 4 depicts the effect of using resources to model rules that govern duty time and work time on a DB pairing problem. The input is a subset of the daily long-haul conductor problem, and contains 1,361 legs. We see that the $k$-shortest path approach without resources is unable to produce acceptable solutions for this problem. The reason is that the shortest path enumeration frequently times out after finding 10,000 paths, all of which are illegal. We also note that the resource merging technique not only reduces the runtime, but also improves the quality of the solution. This can be explained by the fact that the number of labels created by the resource-constrained shortest path solver may be limited by memory consumption when merging is not used.

## 6.6 Dual Strategy

As discussed in Sect. 4.2, one of the tasks of the master optimizer is to provide duals for the column generator. One approach to implementing the master optimizer is to relax the integrality conditions from the integer program, and solve the resulting LP, either using a simplex solver, or an interior point solver without crossover. A simplex solver returns values corresponding to extreme points of the optimal face, whereas interior point solutions are in the relative interior of the optimal face, which is usually a better representative of the possible dual solutions (Lübbecke and Desrosiers, [10]).

Another approach is to Lagrangian relax all the constraints and solve the relaxation using a subgradient approach. The details of this, along with a dual ascent approach to find integer solutions, are presented in Gustafsson ([6]). In brief, the integer heuristic and subgradient solver is highly integrated, and is run continuously with every call to the pricing module. A high focus is put on keeping the computational effort very low, and for example during the early stages of a complete column generation run, we accept sub-optimal subgradient duals as long as the integrality gap is reasonable.

Fig. 5 shows a comparison of these three approaches on a 530 leg problem. Fig. 5a tracks the progress of the lower bound with all integrality restrictions

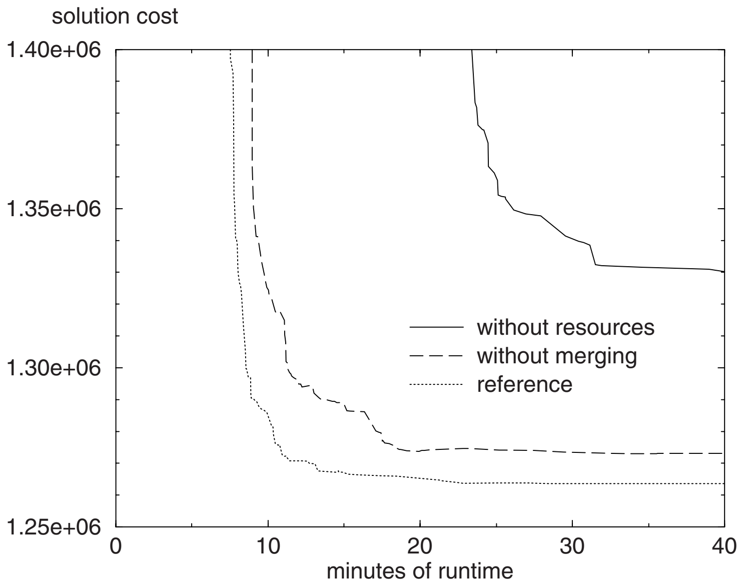
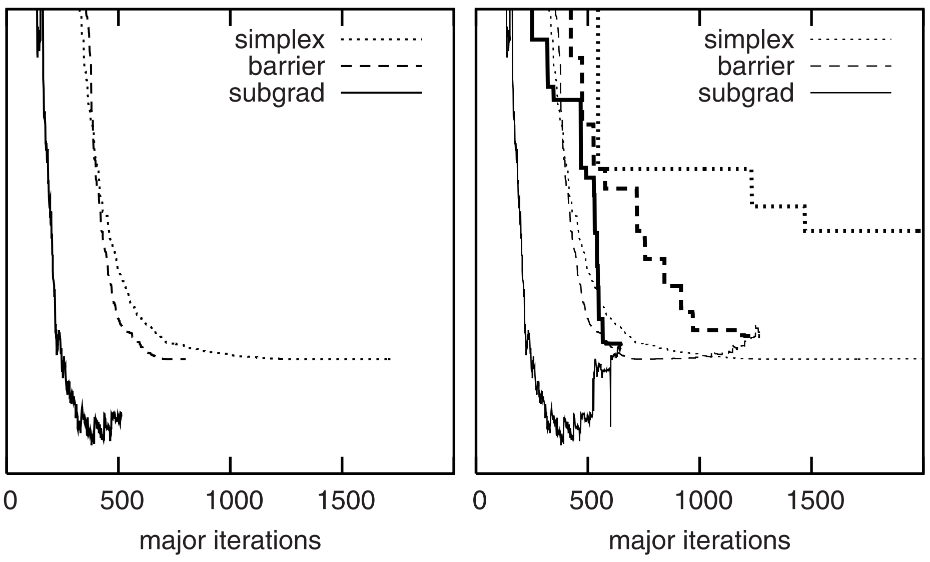


**Fig. 4.** Comparison of column generation runs. The reference run uses both resources and a label merging technique. None of the runs showed any further significant improvements after 40 minutes of runtime on a 2.8 GHz Pentium-4 server with 2 GB of memory.

removed. As expected, the LP solvers converge to the same lower bound, whereas the subgradient solver provides a significantly weaker lower bound (in this case about 0.5% lower). However, the simplex approach has a very long tail, requiring 1,800 iterations to reach convergence, whilst the interior point approach requires 800 iterations, and the subgradient only 500.

In Fig. 5b we also show the upper bounds coming from the dual ascent heuristic. To reduce the gap between the upper and lower bounds, we have implemented a simple integrality strategy. When no more negative reduced cost columns exist, a number of leg-leg connections are fixed to one, and the column generation process continues. The fix-and-regenerate steps continue until the lower bound to the integer restricted problem is fathomed by the best known solution. The upper bound converges fastest for the subgradient approach, and slowest for the simplex case. In fact the graph truncates the simplex run; convergence is reached after 5,000 major iterations, and then to a poorer quality solution.

Whilst Fig. 5 applies to one particular problem, the results are similar for other problems. As the size of the pairing problem grows, the difference between the approaches becomes even more significant. A further advantage of the subgradient approach is that the computations are much quicker. For the 530 leg problem, a subgradient major iteration is on average twice as fast as an interior point iteration, and four times faster than a simplex major iteration.



(a) Lower bound for the RMP till convergence

(b) LB (thin lines) and UB (thick lines) with connection fixing

**Fig. 5.** The effect of different master optimizers

## 6.7 Integer Strategy

The dual-ascent heuristic that provides integer solutions to the master problem was devised by Wedelin ([14]). This solver is fully integrated with the generation step, and typically finds new solutions every two or three pricing calls. Due to its probabilistic nature, not every new solution is the globally best so far; still, it is a very dynamic solver that produces close to optimal solutions throughout the run. This gives the end-user the ability to stop the run early (before full convergence), if the solution is deemed "good enough".

As mentioned in the previous section, a connection fixing strategy aids the search for high quality integer solutions. Starting with a fractional solution to the integrality relaxed master problem, we determine fractional leg-leg connections, and fix the connections that are closest to one. Columns and network arcs that violate fixed connections are removed, at least temporarily. When solving very large pairing problems ($> 2,000$ constraints), the time taken to calculate the fractional solution can be very large if standard LP solvers are used. Instead we have implemented an approximate solver, which is similar to the volume algorithm of Snowdon et al. ([12]).

We have further improved the connection fixing strategy by implementing *early branching* and by unlocking some or all connections when the lower and upper bounds meet, and then continuing. In early branching we no longer wait for LP convergence, but instead fix connections after a certain number of pricing iterations have passed. This has the effect of returning high quality integer solutions early in the search history, and can even reduce overall run time. If desired, a valid lower bound can be determined after the early branching by unlocking all connections, and generating columns until convergence is reached. Fig. 6 shows the effectiveness of this approach. In both cases an interior point solver was used as the LP optimizer. Each line shows the primal objective averaged over eight runs, each with a different starting solution produced by a very simple construction heuristic.

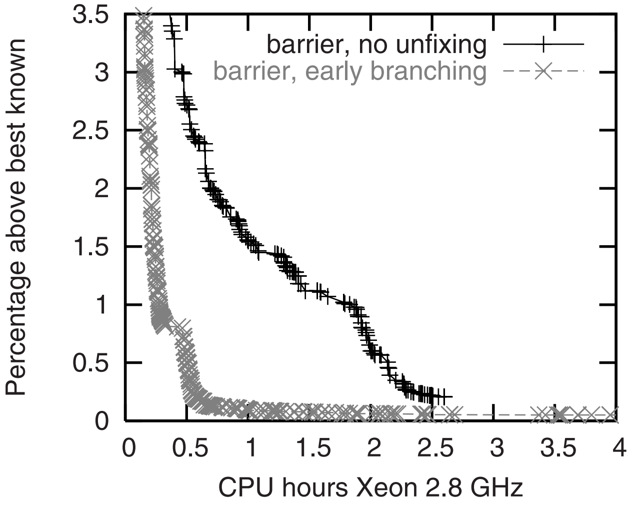


**Fig. 6.** Different integrality strategies

## 7 Large-Scale Results

We now integrate the new algorithmic contributions, and see how the resulting pairing optimizer performs on a range of railway problems. As a challenging test case, we have chosen the DB long distance planning problem, encompassing close to 7,000 daily legs and many more deadhead possibilities. There are 24 crew bases, 211 stations, and a number of different types of base constraints (Sect. 5.2), strongly affecting the nature of the solutions. Fig. 7 shows the results. We are able to solve the full problem in less than a day on standard PC hardware. The smaller problems presented in the graph were created by taking the solution from the full run, and selecting legs operated by a subset of the bases. The bases in the smaller problems are close geographically, making it non-obvious how the legs should be distributed across the bases.

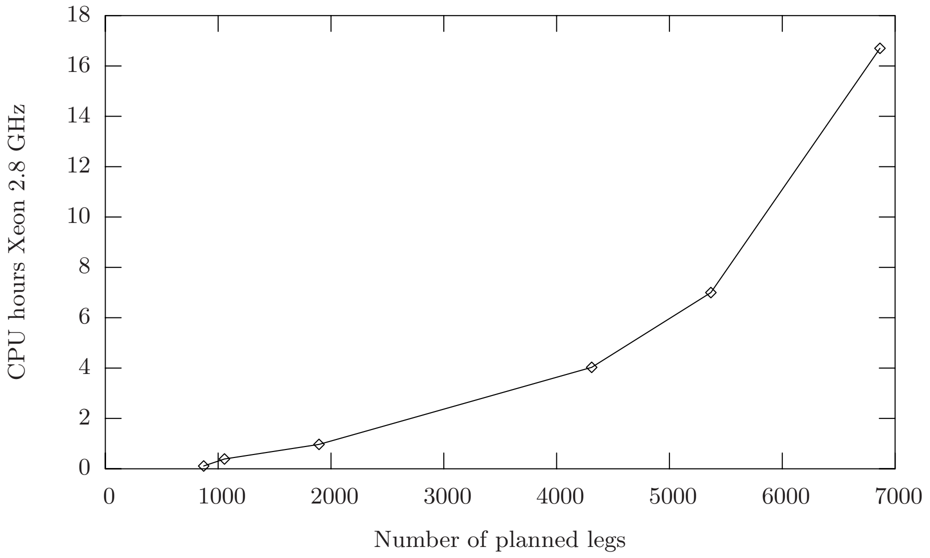


**Fig. 7.** Run times for various problem sizes

## 8 Conclusions and Future Work

Our results illustrate that it is possible to solve large and highly complex railway pairing problems in reasonable time using modern OR techniques. We believe that one of the key factors behind the success of our approach is the black box rule system, which separates the generic core optimization algorithms from the very user-specific problem definition, and makes it possible for planners to express rules and cost components in the tailor-made "Rave" language. An alternative approach, in which certain standard rules are "hard-coded" in the optimizer, would probably not have been flexible enough, given the complexity of the DB rule and cost structure. Another key factor is the column generation approach, which provides near-optimal solutions to the mathematical formulation of the pairing problem. Compared to manual planning, we have seen cost savings of 10–15% on large railway problems from several operators.

There are some aspects of the pairing optimizer that we are currently working on. Optimization speed is one of them, of course. Another is to improve the transportation leg search in the pricing problem. It is not feasible to insert all buses and trains in the whole region into the pricing network, especially not if the planning period stretches over a month. Finding the "right" set of transportation connections a priori is a non-trivial problem, however. Preferably the optimizer should find the right deadheads without manual intervention. Various preprocessing techniques can be used for this purpose.

## Acknowledgements

We would like to thank our colleague Stefan Karisch, who helped in coordinating this work, and to the referees for their valuable comments.

## References

1. Azevedo, J., Martins, E.: An algorithm for the multiobjective shortest path problem on acyclic networks. Investigacao Operacional 11(1), 52–69 (1991)
2. Caprara, A., Fischetti, M., Toth, P., Guida, P.L., Vigo, D.: Algorithms for railway crew management. Mathematical Programming B 79, 125–141 (1997)
3. Caprara, A., Fischetti, M., Toth, P., Vigo, D., Guida, P.L.: Solution of large-scale railway crew planning problems: the Italian experience. In: Wilson, N. (ed.) Computer-Aided Transit Scheduling. Lecture Notes in Economics and Mathematical Systems, pp. 1–18. Springer, Heidelberg (1999)
4. Cordeau, J.-F., Toth, P., Vigo, D.: A survey of optimization models for train routing and scheduling. Transportation Science 4, 380–404 (1998)
5. Desaulniers, G., Desrosiers, J., Dumas, Y., Marc, S., Rioux, B., Solomon, M.M., Soumis, F.: Crew pairing at Air France. European Journal of Operations Research 97, 245–259 (1997)
6. Gustafsson, T.: A Heuristic Approach to Column Generation for Airline Crew Scheduling. Lic. thesis, Chalmers University of Technology, Gothenburg, Sweden (1999)
7. Hjorring, C., Hansen, J.: Column generation with a rule modelling language for airline crew pairing. In: 34th Annual Conference of the Operational Research Society of New Zealand, pp. 133–142 (1999)
8. Kroon, L., Fischetti, M.: Crew scheduling for Netherlands Railways, Destination: Customer. In: Voss, S., Daduna, J. (eds.) Computer-Aided Scheduling of Public Transport. Lecture Notes in Economics and Mathematical Systems, pp. 181–201. Springer, Heidelberg (2001)
9. Lavoie, S., Minoux, M., Odier, E.: A new approach for crew pairing problems by column generation with an application to air transportation. European Journal of Operations Research 35, 45–58 (1988)
10. Lübbecke, M.E., Desrosiers, J.: Selected topics in column generation. Technical report, GERAD, Montreal, Canada (2002)
11. Minoux, M.: Column generation techniques in combinatorial optimization: A new application to crew pairing problems. In: XXIVth AGIFORS Symposium (1984)
12. Snowdon, J., Anbil, R., Pangborn, G.: The airline crew scheduling problem: Dual simplex, volume, and volume/sprint solutions. Technical report, IBM, T.J. Watson Research Center (2000)
13. Vance, P.H., Barnhart, C., Johnson, E.L., Nemhauser, G.L.: Airline crew scheduling: A new formulation and decomposition algorithm. Operations Research 45(2), 188–200 (1997)
14. Wedelin, D.: An algorithm for large scale 0-1 integer programming with application to airline crew scheduling. Annals of Operations Research 57, 283–301 (1995)
15. Wren, A., Fores, S., Kwan, A., Kwan, R., Parker, M., Proll, L.: A flexible system for scheduling drivers. Journal of Scheduling 6, 437–455 (2003)