Improved Search for Night Train Connections

Thorsten Gunkel, Matthias Müller–Hannemann and Mathias Schnee

Darmstadt University of Technology, Computer Science,
64289 Darmstadt, Hochschulstraße 10, Germany
`muellerh,schnee@algo.informatik.tu-darmstadt.de`,
`http://www.algo.informatik.tu-darmstadt.de`

Abstract. The search for attractive *night train connections* is fundamentally different from ordinary search: the primary objective of a costumer of a night train is to have a reasonably long sleeping period without interruptions due to train changes. For most passenger it is also undesired to reach the final destination too early in the morning. These objectives are in sharp contrast to standard information systems which focus on minimizing the total travel time.
In this paper we present and compare two new approaches to support queries for night train connections. These approaches have been integrated into the Multi-Objective Traffic Information System (MOTIS) which is currently developed by our group. Its purpose is to find all train connections which are attractive from a costumer point of view.
With a computational study we demonstrate that our specialized algorithms for night train connections are able to satisfy costumer queries much better than standard methods. This can be achieved with reasonable computational costs: a specialized night train search requires only a few seconds of CPU time.

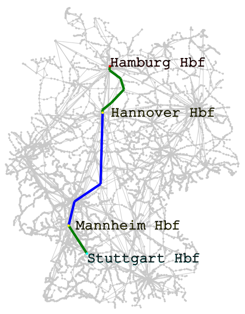
Keywords: timetable information system, multi-criteria optimization, night trains, computational study

1 Introduction and Motivation

Marketing campaigns of major railway companies praise the advantages of night trains: "By traveling at night you save paying a hotel night, and you gain a full day of activities." Compared to traveling by plane, passengers can take more luggage with them, and they save the check-in procedures at airports and transfers from the airport to the city center.

At a first glance, it may seem surprising that the same railway companies spend only little effort to support potential customers in their search for attractive night train connections. However, we will explain later in this paper why an efficient night train search is computationally quite challenging.

Current search engines either do not support an explicit search for night trains at all or their functionality is quite limited. The latter type of search engines supports only direct connections and requires that the user already knows from which night train station he wants to start and at which night train station



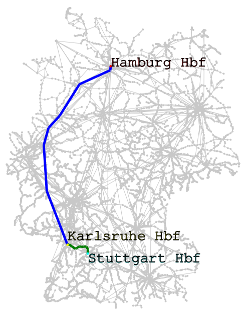


Fig. 1. Example: Alternative night train connections from Stuttgart Hbf to Hamburg Hbf.

he wants to leave. Of course, the search of direct connections is algorithmically very simple. The problem immediately becomes much more difficult if the starting point or the final destination are not served by a night train connection at all. In general, there will be several night train stations in the neighborhood of the starting point and the destination of a journey which has to be planned. Thus, this paper deals with a complex environment of a relatively dense network (like the railway network of central Europe) which offers many alternatives. The goal of this paper is to introduce and to discuss several approaches for an effective night train search for such a scenario.

In general, we look for a connection consisting of three parts (the first and third part may be empty):

– one or more feeder trains from the origin to the entry point of a night train,
– a night train, and
– again one or more feeder trains from the station exit point of the night train to the final destination.

The purpose of the initial feeder trains is to bring the costumer in time (with a certain safety margin) to the night train. For the feeder trains (in the first and the third part), we aim for fastest and most convenient connections with respect to the number of interchanges, whereas the night train section should have a minimum length of h hours. The parameter h can be set by the costumer, a typical choice might be $h = 6$ hours.

Thus, the overall connection which we are looking for will typically not be the fastest possible, and that is why information servers which focus on fastest connections will fail to find and offer them. If there are several alternatives for the arrival time at the destination, the search engine should present all alternatives. Fig. 1 shows an example of a query from Stuttgart Hbf to Hamburg Hbf with two alternative night train connections. The first connection is faster with a total duration of 8 h 23 min, but requires two train changes and has a sleeping period of only 5 h 19 min. The second connection has a total duration of 9 h 54 min, only one train change but offers an uninterrupted sleeping period of 8 h 02 min.

Related work. In recent years, there has been strong interest in efficient algorithms for timetable information. Two main approaches have been proposed for modeling timetable information as a shortest path problem: the *time-expanded* [1,2,3,4,5], and the *time-dependent* approach [6,7,8,9,10,11,12,5]. The common characteristic of both approaches is that a query is answered by applying some shortest path algorithm to a suitably constructed graph. These models and algorithms are described in detail in a recent survey [13].

Several recent publications on timetable information systems focus merely on performance issues to find fastest connections, and mostly consider only greatly simplified single criteria scenarios. These simplified models ignore aspects like days of operation, transfer times and restrictions, desired train attributes, meta stations, footpaths between stations, just to name a few.

Multi-criteria search for train connections in a realistic environment has been studied in [4]. In this paper, we adopt the same philosophy: our underlying model has to ensure that each proposed connection is indeed feasible, that is, can be used in reality by a potential costumer. Moreover, our focus is on the quality of the proposed connections and we aim at presenting attractive alternatives to customers.

Our Contribution. We are not aware of any previous work on night train search. Our first contribution in this paper is a formal model which tries to capture the notion of attractive night train connections. In Section 2, we first review the notion of relaxed Pareto optimality from [4]. Afterwards, we discuss how to model that a connection offers enough sleeping time and what other aspects should be considered.

Based on this formal model, we develop two general approaches for night train search. The first approach is an enumerative approach. It is based on the idea that there are only relatively few night trains which are candidates for a given query.

Our second approach considers sleeping time as an additional criterion in a multi-criteria search. Here we extend a multi-criteria version of Dijkstra's algorithm to this additional criterion.

The basic versions of both general approaches are quite inefficient. Therefore, we have engineered both of them. By using appropriate speed-up techniques we achieve acceptable average running times of only a few seconds per query. In an extensive computational study we show that our fastest versions yield high quality solutions, much better than what we can reach by standard methods.

Overview. The rest of the paper is organized as follows. We start with our formalization of attractive night train connections, followed by a brief description of MOTIS in Section 3. Then, we introduce two general approaches to night train search in Section 4. Afterwards we present computational results based on a large test set of real customer queries. Finally, we conclude with a short summary.

2 Attractive Night Train Connections

2.1 General Considerations

A simple measurement for the "attractiveness" of a connection does not exist. Different kinds of costumers have differing (and possibly contrary) preferences. Key criteria for the quality of a connection are travel time, ticket cost and convenience (number of interchanges, comfort of the used trains, time for train changes). In order to build a traffic information system that can provide attractive connections we avoid the drawbacks of weighted target functions or "preference profiles". Instead we want to serve each possible costumer by presenting him a selection of highly attractive alternatives with one single run of the algorithm.

When dealing with multiple criteria a standard approach is to look for the so-called Pareto set. For two given k-dimensional vectors $x = (x_1, \ldots, x_k)$ and $y = (y_1, \ldots, y_k)$, x *dominates* y if $x_i \leq y_i$ for $1 \leq i \leq k$ and $x_i < y_i$ for at least one $i \in \{1, \ldots, k\}$. Vector x is *Pareto optimal* in set X if there is no $y \in X$ that dominates x. Here, we assume for simplicity that all criteria shall be minimized. It should be obvious how these definitions have to be adapted if some criterion has to be maximized.

We argued in [4] that the set of Pareto optima still does not contain all attractive connections and proposed to apply the concept of *relaxed Pareto optimality*. It provides more alternatives than Pareto optimality can give. Under relaxed Pareto dominance

- connections that are nearly equivalent but differ slightly do not dominate each other;
- the bigger the difference in time between start or end of two connections the less influence they have on each other.

We use the following rules to compare connections A and B which have departure times d_A, d_B, arrival times a_A, a_B, travel times t_A, t_B (all data given in minutes) and i_A, i_B interchanges, respectively. Connection A *dominates* connection B

- with respect to the criterion travel time if B does not overtake A and

$$t_A + \alpha(t_A) \cdot \min\{|d_A - d_B|, |a_A - a_B|\} + \beta(t_A) < t_B,$$

  where, $\alpha(t_A) := t_A/360$ and $\beta(t_A) := 5 + \sqrt{t_A}/4$;
- with respect to the number of interchanges only if $i_A < i_B$;

For ease of exposition we omit in this paper further rules which consider ticket costs. The interested reader is referred to [14].

2.2 Discussion of Objectives for Night Trains

How can we ensure that a connection offers enough sleeping time? From a modeling point of view, we could simply impose a lower bound on the sleeping time as a side constraint. Let us call this lower bound *minimum sleeping time* and denote its value by lb_{st}.

Unfortunately, the choice of some suitable constant lb_{st} is not obvious since different customers may have very different opinions on what they regard as sufficient sleeping time. But even if customers are allowed to choose this constant individually according to their personal preferences, any sharp border imposed by such a constant is questionable. If we choose lb_{st} too large we may miss valuable alternatives (which are just below the given value). In contrast, choosing the constant lb_{st} too small may lead to relatively short sleeping periods, since the search algorithm has no incentive to favor alternatives with longer sleeping periods.

However, to use the pure objective "maximize the sleeping time" is also questionable as it supports unnecessary, but costly detours. Thus, we have to balance the goal to maximize the sleeping time with the usual goal to minimize the overall travel time.

Therefore, we combine both ideas and propose the following model. We choose a fairly small lower bound on the minimum sleeping time, to distinguish night train connections which include a reasonable sleeping period from other connections which only partially use a night train.

Suppose we want to compare two connections c_1 and c_2 with total travel times $tt(c_1)$ and $tt(c_2)$ and sleeping times $st(c_1)$ and $st(c_2)$, respectively. We suggest the following domination rules:

1. If connection c_1 is faster than c_2, then the increase in sleeping time $st(c_2) - st(c_1)$ should be at least as large as the increase in total travel time $tt(c_2) - tt(c_1)$. Otherwise, we consider c_2 as dominated by c_1 with respect to these two criteria.
2. We also impose an upper bound on the sleeping time ub_{st}. The idea is that sleeping times longer than this upper bound should not be considered as beneficial for the customer. Thus, instead of using the original sleeping time st, we use a *modified sleeping time* $mst := \min\{st, ub_{st}\}$ in our comparisons of connections.

2.3 Filtering Attractive Solutions

Trains are considered as *night trains* if they are officially labeled as such (and not just operate during the night). A connection is considered as a *night train connection* only if it includes a night train with a sleeping time of at least lb_{st} minutes.

This definition does only partially capture what passengers will consider as an *attractive* night train connection. Therefore, we propose to apply additional criteria to reduce the result sets further. In this paper, we use the following additional rules:

– We remove all night train connections with an extremely long feeder section, since such connections usually imply a large detour. To this end, we use an upper bound on feeder lengths ub_{fe}.

– We also remove all connections which have more than two additional interchanges than some other night train connection as such connections are quite uncomfortable.
– From the remaining solutions, we filter out all dominated solutions, where we use modified sleeping time $mst := \min\{st, ub_{st}\}$ as explained above.

Since ticket costs depend very much on the chosen train category and the fare system is quite complicated, we do not consider ticket costs in this paper for ease of exposition.

3 The Information Server MOTIS

This section is intended to give a brief introduction to MOTIS and the main ideas behind it. In the following subsections we first explain what kind of queries can be handled. Afterwards we briefly touch upon the graph model used and the general search algorithm.

3.1 Queries

A *query* to a timetable information system usually consists of the *start station* (or origin) of the connection, the *terminal station* (destination) and an *interval* in time in which either the departure or the arrival of the connection has to be, depending on the *search direction*, the user's choice whether to provide the interval for departure ("forward search") or arrival ("backward search"). If several stations are relatively close together, they are grouped together to form virtual *meta-stations*. The search engine treats all stations belonging to the same meta-station as equivalent. Additional query options include:

Train class restrictions. Each train has a specific *train class* assigned to it. These classes are high-speed trains such as the German ICE and French TGV; ICs and ECs and the like; local trains, "S-Bahn" and subway; busses and trams. The *query* may be restricted to a subset of all *train classes*.

Attribute requirements and night train categories. Trains have *attributes* describing additional services they provide. Such attributes are for example: "bike transportation possible" or"board restaurant available". Night trains offer different categories, for example reclining seats, couchettes (unisex sleeping compartments), or sleepers (private and comfortable sleeping accommodation available as singles, doubles or triples). Users who wish to have a minimum standard of comfort can specify which night train categories are acceptable for them. The default specification in night train search is to accept all night train categories.

3.2 Time-Expanded Graph Model

The basic idea of a so-called *time-expanded graph model* is to introduce a directed search graph where every node corresponds to a specific event (departure, arrival, change of a train) at a station.

A connection served by a train from station A to station B is called *elementary*, if the train does not stop between A and B. Edges between nodes represent either elementary connections, waiting within a station, or changing between two trains. For each optimization criterion, a certain length is associated with each edge.

Traffic days, possible attribute requirements and train class restrictions with respect to a given query can be handled quite easily. We simply mark train edges as *invisible* for the search if they do not meet all requirements of the given query. With respect to this visibility of edges, there is a one-to-one correspondence between feasible connections and paths in the graph.

More details of the graph model can be found in [4].

3.3 The Search Algorithm in MOTIS

Our algorithm is a "Pareto-version" of Dijkstra's algorithm using multi-dimensional labels. Pseudocode is given in Algorithm 1. See Möhring [15] or Theune [16] for a general description and correctness proofs of the multi-criteria Pareto-search. In this algorithm, each label is associated with a node v in the search graph. A label contains key values of a connection from a start node up to v. These key values include the travel time, the number of interchanges, a ticket cost estimation and some additional information. For every node in the graph we maintain a list of labels that are not dominated by any other label at this node. In the beginning, all label lists are empty.

Then, start labels are created for all nodes with a timestamp within the query interval and stored in a priority queue (lines 5-7). In the main loop of the algorithm, one label is extracted from the priority queue in each iteration (line 9). For the corresponding node of that label all outgoing edges are scanned and labels for their head nodes are created, provided that the edge is feasible (lines 10-12). Any new label is compared to all labels in the list corresponding to its node. It is only inserted into that list and into the priority queue if it is not dominated by any other label in the list. On the other hand, labels dominated by the new label are removed (line 18).

As a further means of exploiting dominance we keep a short list of Pareto-optimal labels at the terminal station (called `topTerminalLabelList`) and compare each new label to these labels (line 14). To compare labels at an intermediate node v with a node at the terminal, we use lower bounds on the key values of a shortest, a most convenient, and a cheapest path from v to the terminal station. We increase the criteria of the label at v by lower bounds on the according values. If the label with its increased values is dominated by any label at the terminal, it is excluded from further search.

Since this optimization can only work with at least one label at the terminal station, we initially determine a guaranteed fastest connection from source to target using a goal-directed single criterion search in an initialization phase before the actual multi-criteria search. This search is by orders of magnitude faster than the multi-criteria search and can be performed in less then 50ms on average.

```
Input: a timetable graph and a query
Output: a set of Pareto-optimal labels at the terminal
foreach node v do
    list<Label> labelListAt(v) := $\emptyset$;
list<Label> topTerminalLabelList := $\emptyset$;
PriorityQueue pq := $\emptyset$;
foreach node v in start interval do
    Label startLabel := createStartLabel(v);
    pq.insert(startLabel);
while ! pq.isEmpty() do
    Label label := pq.extractLabel();
    foreach outgoing edge e=(v,w) of v=label.getNode() do
        if isInfeasible(e) then continue; // ignore this edge
        Label newLabel := createLabel(label, e);
        if newLabel is dominated by labelListAt(w) then continue;
        if newLabel is dominated by topTerminalLabelList then continue;
        // newLabel is not dominated
        pq.insert(newLabel);
        labelListAt(w).insert(newLabel);
        labelListAt(w).removeLabelsDominatedBy(newLabel);
        if newLabel qualifies for topTerminalLabelList then
            topTerminalLabelList.insert(newLabel);
```

Algorithm 1: Pseudocode for the generalized Dijkstra algorithm.

4 Approaches for Night Train Search

In this section we describe two new approaches which we have developed for night train search.

4.1 Pre-Selection of Night Trains

We first present an enumerative approach. Its general idea is to select suitable night train sections first, and then to compute corresponding feeder sections. The main steps can be stated quite easily.

1. Iterate over all night trains of the train schedule which operate on the query day.
2. For each such train, determine all stations which may serve as entry point and all stations which may serve as exit points.
3. For each such pair, determine feeder sections to compose complete connections.
4. Let $\mathcal{C}$ be the collection of connections determined. Apply Pareto dominance to filter out all dominated connections from $\mathcal{C}$. Return the result.

In the following we will first describe steps 2 and 3 in more detail, afterwards we will discuss how to speed up this general approach.

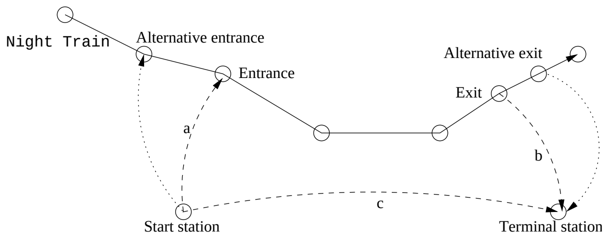


Fig. 2. Selection of pairs of entry and exit points. Pairs are rejected if $a+b > \alpha\cdot c$, i.e., if they would induce a too large detour.

Selection of Entry and Exit Points. Given a query and a particular night train, we have to select in step 2 suitable pairs of entry and exit points to this train. This has to be done with care to achieve a reasonable efficiency. Thus in this phase we intend to reject as many pairs as possible without loosing valuable solutions.

A station where a night train stops (and boarding/deboarding is allowed) qualifies as a possible entry or exit point if it is close with respect to some distant metric to the start or to the terminal station of the query, respectively.

To this end, two metrics can be used: Euclidean distance and lower bounds on the travel time for the feeder section. The advantage of Euclidean based bounds is that we can compute them in constant time. However, such bounds ignore completely the railway network and the train schedule. Two stations which are geographically close may be far from each other with respect to public transport. Estimates on the required travel time between two stations would allow to make more accurate decisions. We propose to use lower bound on the travel time as estimates. These bound can be computed quite efficiently.

As the length of required feeder sections depends very much on the given query, we do not use any fixed absolute bound to decide whether two stations are close enough to each other. Instead we propose to use a query-dependent rejection rule which is visualized in Fig. 2. A pair of entry and exit points is rejected for a query if the bound a on the feeder length from the start station to the entry point and the bound b on the feeder length from the exit point to the terminal station together exceed the bound c on the length of a direct connection between start and terminal station by some factor α, i.e., if

$$a + b > \alpha \cdot c.$$

Our experiments revealed that setting $\alpha := 1$ is a suitable conservative choice.

Finally, we accept a pair of entry and exit stations only if the travel time of the corresponding night train between these two stations is above our lower bound on the sleeping time lb_{st}.

Computation of Feeders. Given a pair of entry and exit points for a night train the next step is to compute feeder trains.

The entry point for the night train determines when we have to arrive at this particular station at the latest. Since we really want to reach the night train we incorporate some extra safety margin to this calculation. Then we can use an ordinary backward search from this station and the latest arrival time to the start station to find suitable feeder trains.[1] Likewise we perform an ordinary forward search from the exit point to the terminal station.

Since entry and exit points are likely to appear in several pairs, we have to make sure not to compute the same feeder sections several times. To avoid repeated calculations, we therefore introduced a caching mechanism which stores the results of each feeder search.

Pruning the Search Space. A naive implementation of our enumerative approach would do the feeder computation in an arbitrary order for all selected pairs. Since the selection of pairs is done in a very conservative way, the resulting algorithm would be quite inefficient.

A more clever refinement of this approach uses a priority queue to determine the order of feeder computations. The idea is that already computed solutions can be used to prune the search space. The priority queue contains all pairs for which at least one feeder has not been computed yet. The key by which we order the entry and exit point pairs in the priority queue is an estimate on the travel time of the overall connection. This travel time estimate is composed by the known length of the night train section plus estimates on the feeder lengths. When a particular feeder has been determined during the course of the algorithm, our estimates are updated for all elements in the priority queue where this feeder fits. In each iteration we select and remove the top element from the priority queue. For the corresponding pair we check whether it is already dominated by previously computed connections. If this is the case, we discard this pair. Otherwise, we compute one missing feeder. Afterwards we either obtain a set of complete connections for this pair, or the other feeder section is still missing. In the latter case, we reinsert the pair into the priority queue with the updated key information.

4.2 Multi-Criteria Search with an Additional Criterion

The second approach which we propose adds sleeping time as a new criterion to the multi-criteria search for attractive connections. Form a software-engineering point of view the multi-criteria framework implemented in MOTIS is easily extendable to an additional criterion. In general, only two modifications are necessary.

1. We have to make sure that the labels representing partial connections keep track of the additional criterion.

[1] Ordinary search allows the replacement of start and terminal stations by equivalent meta-stations. The possibility for such a replacement has to be switched off for the entry and exit point as in our scenario we really have to arrive at the pre-selected station and not at some equivalent one.

2. The domination rules have to be adapted so that they effectively prune labels.

While the modification of labels is straightforward, finding good domination rules is much more difficult (and usually requires some experimental evaluation).

Pruning of labels during search by domination can only be done with the help of good and efficiently computable bounds, lower bounds for minimization and upper bounds for maximization, respectively.

Thus, for the maximization criterion sleeping time we need an upper bound. Given a partial connection, this bound should limit the maximum additional sleeping time this connection can accumulate to the terminal station. With the help of such an upper bound a label of a partial connection can be dominated with respect to the criterion sleeping time if the current sleeping time plus the additional sleeping time is smaller than the sleeping time of some known complete connection. Unfortunately, we do not know such upper bounds, except for trivial ones which are far too loose to help in pruning.

Since a Pareto search without pruning is hopeless (although the search space is polynomially bounded in practice [17], it is still way too large to achieve computation times of a few seconds), we have to use heuristic domination rules which cannot guarantee to find all attractive solutions.

We adapt the domination rules of MOTIS as follows: A complete connection c is only allowed to prune a partial connection p

– if p “has used and already left” a night train but did not reach at least lb_{st} sleeping time, or
– if p “has used and already left” a night train but did not reach more sleeping time than c, or
– if p is currently “in a night train” then c has to have sleeping time above the threshold lb_{st}, and the sleeping time of c has to be at least the sleeping time of p plus β times a lower bound on the remaining travel time for p (for some constant β), or
– if p contains no night train at all.

While the first two rules are still exact, the two others are aggressive heuristics.[2]

If c is allowed to prune it still needs to be “relaxed Pareto smaller” with respect to the other criteria. For the comparison of labels belonging to the same node (i.e., partial connection against partial connection) nothing has to be changed.

5 Computational Results

5.1 Test Cases

We took the train schedule of trains within Germany of July 2007. For our experiments, we used a snapshot of about 25000 real customer queries of Deutsche

[2] Initial experiments showed that without these heuristics the average CPU time would be about one minute. This is clearly not acceptable for on-line use of information systems.

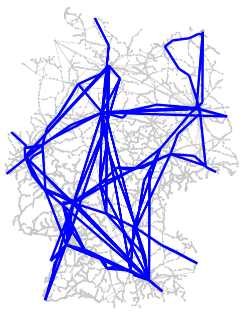


Fig. 3. The railway network of Germany. All night train routes are highlighted.

Bahn AG. From these we selected and processed only those 1782 queries where the straight line distance between start and terminal station was at least 350 km. For all other queries the distance is likely to be too short to allow for a reasonable night train connection.

Among the 1782 queries, we have 347 queries which possess a direct night train connection and 940 require only one feeder. The remaining 495 queries need two feeders. The current schedule and the derived time-expanded graph have sizes as shown in Table 1.

5.2 Specific Definition of Attractive Solutions

We have chosen the following constants to specify our notion of attractive night train connections as introduced in Section 2.

- A connection is considered as a *night train connection* only if it includes a night train with a sleeping time of at least $lb_{st} = 240$ minutes.
- We limit the maximal travel time of some feeder section also to $ub_{fe} := 240$ minutes.
- In our definition of the modified sleeping time $mst := \min\{st, ub_{st}\}$ (as introduced in Section 2) we have chosen the upper bound as $ub_{st} = 420$ minutes.

5.3 Computational Environment

All computations are executed on an AMD Athlon(tm) 64 X2 dual core processor 4600+ with 2.4 GHz and 4 GB main memory running under Suse Linux 10.2. Our C++ code has been compiled with g++ 4.1.2 and compile option -O3.
We compare the following variants:

- Algorithm A: our standard MOTIS version which was designed to find all attractive train connections with respect to travel time minimization and minimizing the number of train interchanges. MOTIS requires a time interval specifying when the connection has to start. To use MOTIS for a night

number of stations	8 916
number of trains	56 994
number of night trains	229
number of nodes	2 400 534
number of edges	3 715 557

Table 1. Key parameters of the schedule and the corresponding graph.

train search, we set this start interval to a period between 6:00 pm on the traffic day and 2:00 am on the following day. For our comparison with other variants, we considered only night train connections.
- Algorithm B: the enumerative approach of pre-selecting night trains as described in Section 4.1.
- Algorithm C: a heuristic version of Algorithm B. We replace the multi-criteria search for feeders by a single-criteria search with respect to travel time. The latter is much more efficient, but may lead to additional interchanges. The idea behind this variant is that feeder connections should in general not be very complicated.
- Algorithm D: the multi-criteria version of MOTIS with sleeping time as an additional criterion as described in Section 4.2.

5.4 Experiments

Experiment 1. In our first experiment we want to study the basic question: How often is it necessary to use a specialized night train search to find any suitable night train connection?

To answer this question we compared Algorithm A with all other variants, see Table 2. Algorithm A (standard MOTIS) does not find any night train connection in 370 out of 1782 test cases (20.75%), whereas Algorithms B and C always found at least one reasonable night train connection. This already shows that a specialized night train search can offer much more to customers. Our version of Algorithm D (MOTIS with one additional criterion) fails to find a night train connection in 41 cases (2.3%). This is due to our heuristic version of domination rules.

Experiment 2. How does the quality of the result sets compare to each other?

The comparison of the result sets in a multi-objective search space can be done in several ways. A first, but only rough indicator is the size of the solution set after filtering out dominated solutions. The largest result set is delivered by Algorithm B (4223 solutions over all instances), followed by Algorithm C (3939 solutions) and Algorithm D (3196 solutions). Algorithm A delivers only 2334 solutions.

Next we studied which algorithmic variant was able to find the most attractive connection. For this comparison we introduced a quality measure which allows us to rank the solutions for each query.

Algorithm	# connections	CPU time	# failures	
A (standard MOTIS)	2334	1.87s	370	20.75 %
B (pre-selection+feeder)	4223	14.20s	0	0 %
C (pre-selection+fast feeder)	3939	3.72s	0	0 %
D (MOTIS with additional criterion)	3196	2.34s	41	2.3 %

Table 2. The total number of connections found, average running times in seconds, and the number of failures for all variants.

Given a connection c with travel time $tt(c)$ in minutes, modified sleeping time $mst(c)$ also in minutes, and number of interchanges $ic(c)$, we measure the cost of c by the function

$$q(c) := tt - mst + k \cdot ic,$$

where we set the constant $k := 20$ and $ub_{st} = 420$ minutes. The smaller the cost value, the better we regard the quality of the corresponding connection. Our cost function can be interpreted as follows: We have to pay for each minute of travel time. This cost can be reduced by the sleeping time up to our upper bound ub_{st}. An interchange is counted as 20 minutes extra travel time. We now rank the solutions as follows: A direct night train connection has always first rank. All other connections are ranked according to increasing cost. We have experimented with different constants in our cost function. It turned out that the ranking of our algorithms is quite robust against changes it these values.

With respect to this ranking of solutions, we now compared the quality of the first rank solutions against each other. Table 3 shows how often the first ranked solutions have strictly better quality, how often they match, and how often they are strictly worse. We observe that the quality of Algorithm B and Algorithm C is quite similar, whereas Algorithm D has a slightly poorer quality.

Experiment 3. Is their a trade-off between computational efficiency and quality of the solutions?

See Table 2 for the average CPU times for all variants. Standard MOTIS (with an exceptionally long query interval of 8 hours) is the fastest variant with only 1.87 seconds, but fails too often to find a night train connection. Algorithm B which gives the overall best quality is about four times slower than Algorithm C. Since the quality delivered by Algorithm C comes close to that of Algorithm B, it will usually not be worth to use the more expensive Algorithm B.

Algorithm D is slightly faster than Algorithm C, but its quality is also slightly poorer. Thus depending on what is more important either Algorithm D or Algorithm C should be used.

Experiment 4. To gain more insight into the behavior of Algorithms B and C we did some operation counting. The following numbers always represent averages.

From the set of all possible entry and exit points, 1719 have been rejected since they are not served on the query date, from the remaining 1605 entry

B vs. C	# cases
B wins	48
C wins	13
both match	1721

B vs. D	# cases
B wins	317
D wins	229
both match	1220

C vs. D	# cases
C wins	312
D wins	250
both match	1220

Table 3. Pairwise comparison of the first ranked solutions.

points 1144 have been rejected because of our distance criterion, and 1205 pairs were removed because of insufficient sleeping time. We had to calculate 111 feeder sections for each query. This explains why it was crucial to speed up Algorithm B by a more efficient feeder computation. It is worth noting that additional 405 feeder computations have been avoided by our caching mechanism.

6 Conclusions

Our computational study shows that a specialized night train search delivers many more attractive connections than an ordinary search. We have observed a trade-off between quality of the solution sets and computation time. Our implementation of a multi-criteria search with one additional criterion fails to find a good night train connection in a few cases, but is most efficient. The pre-selection approach with a fast feeder computation never failed and delivers almost optimal quality. Both variants are fast enough to be applied in on-line information systems. With additional tuning the running times can probably be reduced further while keeping high quality.

We see two promising perspectives to apply our algorithms in practice. The first one is the scenario for which this paper was written: the user explicitly asks for a night train connection. Then we would recommend to use Algorithm C which delivers an excellent quality. The second scenario is an ordinary query with a start interval in the evening. Then it would be an option to run MOTIS with an additional criterion (Algorithm D) but without spending too much additional computation time. If this search finds attractive night train connections, they can be offered as alternatives to those computed for the query interval.

Acknowledgments

This work was partially supported by the DFG Focus Program Algorithm Engineering, grant Mu 1482/4-1. We wish to thank Wolfgang Sprick for fruitful discussions and close collaboration in the development of MOTIS, and Deutsche Bahn AG for providing us timetable data for scientific use.

References

1. Pallottino, S., Scutellà, M.G.: Shortest path algorithms in transportation models: Classical and innovative aspects. In: Equilibrium and Advanced Transportation Modelling. Kluwer Academic Publishers (1998)

2. Schulz, F., Wagner, D., Weihe, K.: Dijkstra's algorithm on-line: An empirical case study from public railroad transport. ACM Journal of Experimental Algorithmics **5** (2000) Article 12
3. Müller-Hannemann, M., Schnee, M., Weihe, K.: Getting train timetables into the main storage. In: Proceedings of the 2nd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2002). Volume 66 of Electronic Notes in Theoretical Computer Science. Elsevier (2002)
4. Müller-Hannemann, M., Schnee, M.: Finding all attractive train connections by multi-criteria Pareto search. In: Proceedings of the 4th Workshop in Algorithmic Methods and Models for Optimization of Railways (ATMOS 2004). Volume 4359 of Lecture Notes in Computer Science, Springer Verlag (2007) 246–263
5. Pyrga, E., Schulz, F., Wagner, D., Zaroliagis, C.: Efficient models for timetable information in public transportation systems. ACM Journal of Experimental Algorithmics (JEA) **12** (2007) 2.4
6. Cooke, K.L., Halsey, E.: The shortest route through a network with time-dependent internodal transit times. Journal of Mathematical Analysis and Applications **14** (1966) 493–498
7. Orda, A., Rom, R.: Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. Journal of the ACM **37** (1990) 607–625
8. Orda, A., Rom, R.: Minimum weight paths in time-dependent networks. Networks **21** (1991) 295–319
9. Kostreva, M.M., Wiecek, M.M.: Time dependency in multiple objective dynamic programming. Journal of Mathematical Analysis and Applications **173** (1993) 289–307
10. Nachtigal, K.: Time depending shortest-path problems with applications to railway networks. European Journal of Operations Research **83** (1995) 154–166
11. Brodal, G.S., Jacob, R.: Time-dependent networks as models to achieve fast exact time-table queries. In: Proceedings of the 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2003). Volume 92 of Electronic Notes in Theoretical Computer Science. Elsevier (2004) 3–15
12. Pyrga, E., Schulz, F., Wagner, D., Zaroliagis, C.: Towards realistic modeling of time-table information through the time-dependent approach. In: Proceedings of the 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2003). Volume 92 of Electronic Notes in Theoretical Computer Science. Elsevier (2004) 85–103
13. Müller-Hannemann, M., Schulz, F., Wagner, D., Zaroliagis, C.: Timetable information: Models and algorithms. In: Algorithmic Methods for Railway Optimization. Volume 4395 of Lecture Notes in Computer Science., Springer Verlag (2007) 67–89
14. Müller-Hannemann, M., Schnee, M.: Paying less for train connections with MOTIS. In Kroon, L.G., Möhring, R.H., eds.: 5th Workshop on Algorithmic Methods and Models for Optimization of Railways, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2006) <http://drops.dagstuhl.de/opus/volltexte/2006/657>.
15. Möhring, R.H.: Verteilte Verbindungssuche im öffentlichen Personenverkehr: Graphentheoretische Modelle und Algorithmen. In: Angewandte Mathematik - insbesondere Informatik, Vieweg (1999) 192–220
16. Theune, D.: Robuste und effiziente Methoden zur Lösung von Wegproblemen. Teubner Verlag, Stuttgart (1995)
17. Müller-Hannemann, M., Weihe, K.: On the cardinality of the Pareto set in bicriteria shortest path problems. Annals of Operations Research **147** (2006) 269–286