HAL
open science

Recoverable Robust Timetable Information

Marc Goerigk, Sacha Hesse, Matthias Müller-Hannemann, Marie Schmidt, Anita Schöbel

To cite this version:

Marc Goerigk, Sacha Hesse, Matthias Müller-Hannemann, Marie Schmidt, Anita Schöbel. Recoverable Robust Timetable Information. ATMOS - 13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems - 2013, Sep 2013, Sophia Antipolis, France. pp.1–14, 10.4230/OASIcs.ATMOS.2013.1 . hal-00871789

HAL Id: hal-00871789

https://inria.hal.science/hal-00871789v1

Submitted on 10 Oct 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Recoverable Robust Timetable Information *

Marc Goerigk[†1], Sascha Heße[2], Matthias Müller-Hannemann[2], Marie Schmidt[3], and Anita Schöbel[3]

1 **Fachbereich Mathematik**
Technische Universität Kaiserslautern, Germany
goerigk@mathematik.uni-kl.de

2 **Institut für Informatik**
Martin-Luther-Universität Halle-Wittenberg, Germany
sascha.hesse@student.uni-halle.de, muellerh@informatik.uni-halle.de

3 **Institut für Numerische und Angewandte Mathematik**
Georg-August Universität Göttingen, Germany
{m.schmidt,schoebel}@math.uni-goettingen.de

Abstract

Timetable information is the process of determining a suitable travel route for a passenger. Due to delays in the original timetable, in practice it often happens that the travel route cannot be used as originally planned. For a passenger being already en route, it would hence be useful to know about alternatives that ensure that his/her destination can be reached.

In this work we propose a *recoverable robust* approach to timetable information; i.e., we aim at finding travel routes that can easily be updated when delays occur during the journey. We present polynomial-time algorithms for this problem and evaluate the performance of the routes obtained this way on schedule data of the German train network of 2013 and simulated delay scenarios.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory (Graph algorithms, Network problems)

Keywords and phrases timetable information, recoverable robustness, delay scenarios

Digital Object Identifier 10.4230/OASIcs.ATMOS.2013.1

1 Introduction

In timetable information, the following problem is typically considered: Given a timetable, an origin and destination, and an earliest departure time, find the "best" route leading from origin to destination; see [21] for a survey. An obvious criterion to evaluate the quality of a route is its duration (or travel time); however, many other criteria have been suggested, as, e.g., the number of changes or the ticket costs [22, 14, 4]. Also the reliability of a path has been considered as a means to account for delays [14, 20, 22]. In [13], decision trees for passengers' travels under uncertainty are constructed. In a recent work [17, 18], approaches from the field of robust optimization were considered. Robust optimization is an approach to handle uncertainty in optimization problems that dates back to the 70s [24].

* partially supported by grants SCHO 1140/3-1 and MU 1482/4-3 within the DFG programme SPP 1307 *Algorithm Engineering* and by the European Union Seventh Framework Programme (FP7-PEOPLE-2009-IRSES) under grant number 246647 and the New Zealand Government (project OptALI).

† Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-13-1-3066. The U.S Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright notation thereon.

© Marc Goerigk, Sascha Heße, Matthias Müller-Hannemann, Marie Schmidt, and Anita Schöbel; licensed under Creative Commons License CC-BY

13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'13).
Editors: Daniele Frigioni, Sebastian Stiller; pp. 1–14
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

During the late 90s, it received new attention through the work of Ben-Tal, Nemirovski and co-authors [2, 3], that sparked a manifold of concepts and algorithms; among them the Γ-approach of [5], adjustable robustness [1], light robustness [16], or recoverable robustness [19, 25]. In our work we focus on recoverable robustness. This is a two-stage concept: Given a set of recovery algorithms, a solution is considered as being robust when for every scenario it can be "repaired" using an recovery algorithm to become feasible. An application to the uncertain shortest path problem has been considered in [6], where the set of recovery algorithms is given by exchanging up to a constant K arcs of the path. Related work can be found in [23], where a given path is updated to a new solution by either using or removing k arcs. Further applications of recoverable robustness include shunting [10], timetabling [11, 25, 19], platforming [9, 25, 19], the empty repositioning problem [15], railway rolling stock planning [8] and the knapsack problem [7]. In some previous work [17, 18], robust passenger information has been considered. It was shown that finding a strictly robust travel route which hedges against any possible delay scenario is an NP-hard problem and for practical application much too conservative. As an alternative, a robustness concept based on light robustness has been proposed. However, it is assumed that a passengers stays on the planned route whatever happens. In contrast to this, we allow that a passengers changes his/her route even if he/she already started the journey.

Contributions. In timetable information, as in many other problems, the passenger does not know the scenarios from the beginning of his/her trip, but learns the current scenario en route. This aspect has been neglected in previous work. In this paper, we describe a recoverable robustness approach to the timetable information problem which takes into account that the actual scenario is learned at some time point en route, and that the travel route may be updated from this point on. For such a recovery, all possible alternative routes may be chosen. The goal is to include this recovery step in the planning phase, i.e. to find a travel route which may be recovered for every delay scenario from a given uncertainty set.

Furthermore, our approach can deal with complicated delay scenarios, as they occur in public transportation where source delays cause the dropping of transfers and changes in the durations of driving and waiting activities. We develop polynomial-time algorithms that can handle any finite set of scenarios and test them on delay scenarios that are generated by propagating delay in transportation systems.

Using large-scale data modeling the train network of Germany, we show the effectiveness of our approach.

Overview. The remainder of this work is structured as follows: We shortly recapture the nominal timetable information problem, and introduce our recoverable robust model in Section 2. We present a polynomial-time label-setting algorithm in Section 3, and demonstrate its applicability to German railway data provided by Deutsche Bahn AG in Section 4. We conclude the paper and discuss further research directions in Section 5.

2 Model and Notation

2.1 Timetable information

In the following we refer to *train timetables* for the sake of simplicity; however, all results can be transferred to any other type of public transport. The starting point for our considerations is a directed acyclic graph, the so-called *event-activity network* (EAN) $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ which is regarded over a finite time horizon. Nodes $\mathcal{E}$ represent events in the train schedule: They can either be

- arrival events $\mathcal{E}_{arr}$ (modeling the arrival of a certain train at a certain station), or
- departure events $\mathcal{E}_{dep}$ (modeling the departure of a certain train from a certain station).

Events are connected by directed arcs, the *activities*, which can be either

- driving activities $\mathcal{A}_{drive}$ (modeling the trip of a train from a departure event to an arrival event),
- waiting activities $\mathcal{A}_{wait}$ (modeling the time a train spends between an arrival and a departure event for passengers to embark and disembark),
- or transfer activities $\mathcal{A}_{trans}$ (modeling passenger movements from one arrival event to another departure event within the same station).

Each event $i \in \mathcal{E}$ has a schedule time $\pi_i \in \mathbb{N}$; furthermore, to compute how delays spread within this network (see Section 4.2), we may assume that for each activity $(i, j) \in \mathcal{A}$ a minimal duration l_{ij} is known, and thus a buffer time $b_{ij} := \pi_j - \pi_i - l_{ij}$. We assume that the initial timetable π is feasible, i.e., $\pi_j - \pi_i \geq l_{ij}$, hence all buffer times are nonnegative. The timetable information problem consists of finding a path within the event-activity network from one station to another, given an earliest departure time s. More precisely, we introduce two virtual events, namely one origin event u and one destination event v, corresponding to a given origin station s_u and a destination station s_v. The origin event u is connected by origin activities $\mathcal{A}_{org}$ with all departure events at station s_u taking place not earlier than s, while all arrival events at station s_v are connected with v by destination activities $\mathcal{A}_{dest}$. We need to find a path P from u to v in $\mathcal{N}$ such that the nominal travel time $t_{nom}(P) := \pi_{\text{last}(P)} - s$ on P is minimal, where $\text{last}(P)$ denotes the last arrival event on P.

2.2 Delays

Paths with minimal travel time in the EAN may be vulnerable to delays, i.e., in case of delays, the originally planned path may take much longer than planned, or planned transfers may even become infeasible if the connecting train does not wait for a delayed feeder train.

The aim of this paper is to give robust timetable information, i.e., to find paths in the EAN which are less vulnerable to delays. The delays observed in a public transportation system originate from source delays d_a which can occur on the driving and waiting activities a of the train. These delays are partially absorbed by buffer times on the activities, however, they *propagate* through the network to subsequent events along driving and waiting activities and – if a transfer is maintained – along the corresponding transfer activity. We assume that each transfer is assigned a *waiting time* which specifies how long the connecting train will wait for the feeder train. If the delay of the feeder train exceeds the waiting time, the connecting train will depart on time. See Section 4.2 for details on our *delay propagation* method. We denote by $\mathcal{A}_{\text{transfer}}(d)$ the set of maintained transfer activities in scenario d and denote the *delay network* $\mathcal{N}(d) := (\mathcal{E}, \mathcal{A}(d))$ with $\mathcal{A}(d) := \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{transfer}}(d)$. The updated timetable is denoted by $\pi(d)$. In this paper, we make the (simplifying) assumption that at some point in time, the passenger learns about *all* delays and can adapt ('recover') his/her travel route accordingly. We partition the events of the networks in a set U^ξ of events where no delay has occurred so far and the passenger has not learned about future delays and a set V^ξ where he/she knows all delays. We require the following properties of an *information scenario* $\xi = (\mathcal{N}^\xi, \pi^\xi, U^\xi, V^\xi)$ consisting of a delay network $\mathcal{N}^\xi$, a disposition timetable π^ξ on this network, and a partition (U^ξ, V^ξ) of the events $\mathcal{E}$:

- $u \in U^\xi$, $v \in V^\xi$,
- if $\pi^\xi_j > \pi_j$, j is in V^ξ,
- all i with $(i, v) \in \mathcal{A}_{dest}$ are contained in V^ξ,
- if i is in V^ξ, all successors of i are in V^ξ.

A way to define the partition (U^ξ, V^ξ) between nodes U^ξ where no delay information is available and nodes V^ξ with full delay information is to set $U^\xi := \{j : \pi_j < t^\xi\}$, $V^\xi := \{j : \pi_j \geq t^\xi\}$, where t^ξ denotes a *revealing time* $t^\xi \leq \min_{j \in \mathcal{E}: \pi_j^\xi - \pi_j > 0} \pi_j$ for every scenario ξ. For our computational experiments, we obtain $\mathcal{N}(\xi) := \mathcal{N}(d^\xi)$ and $\pi^\xi := \pi(d^\xi)$ by delay propagation, see Section 4.2. However, our methods work for any set of scenarios $\xi = (\mathcal{N}^\xi, \pi^\xi, U^\xi, V^\xi)$ as described above; it is not necessary to know the source delays to apply them. We define the set of activities where scenario ξ is revealed as $\mathcal{A}^\xi := \{(i,j) \in \mathcal{A} : i \in U^\xi, j \in V^\xi\}$. A set of information scenarios will be called an *uncertainty set* and denoted by $\mathcal{U}$. In this paper, we consider only finite uncertainty sets.

2.3 Recoverable Robust Timetable Information

Intuitively, we will call a path P recoverable robust if, when an information scenario ξ occurs while a passenger is traveling on P, this passenger can take a *recovery path* P^ξ, to his/her destination. To formally define recoverable robust paths, we make use of the following observation: Let $\mathcal{U}$ be an uncertainty set and let P be a path from u to v in $\mathcal{N}$.

▶ **Lemma 1.** *For every $\xi \in \mathcal{U}$, P contains exactly one arc from $\mathcal{A}^\xi$.*

We denote this arc by $(i^\xi(P), j^\xi(P))$. We denote by $\mathcal{Q}^\xi(j)$ the set of *recovery paths*, i.e., all paths from a node j to v in $\mathcal{N}^\xi$, and set $\mathcal{Q}^\xi(P) := \mathcal{Q}^\xi(j^\xi(P))$.

▶ **Definition 2.** A path P is called *recoverable robust* (with respect to uncertainty set $\mathcal{U}$) if for any $\xi \in \mathcal{U}$ the set of recovery paths $\mathcal{Q}^\xi(P)$ is not empty.

We assume that the passenger travels on the chosen path P until he/she learns about the information scenario he/she is in, i.e., until node $j^\xi(P)$. Since at this node, the full information of ξ, i.e., $\mathcal{N}^\xi$, π^ξ, U^ξ and V^ξ is revealed to the passenger, he/she can take the best path for this scenario. Thus, we assume that he/she reroutes from his/her current position according to scenario ξ.

The goal of this paper is to find "good" recoverable robust paths. However, there are different ideas on how to measure the quality of a recoverable robust path. We can evaluate

- the *nominal quality*: which recoverable robust path has shortest travel time if no delays occur?
- the *worst-case quality*: which recoverable robust path has the earliest guaranteed arrival time?

Hence, we consider the following bicriteria problem:

▶ **Problem 1.** *Bicriteria recoverable robust paths*
Input: EAN $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ with timetable π, origin u and destination v, starting time s, and uncertainty set $\mathcal{U}$.
Task: Find a path P from u to v in $\mathcal{N}$ which is recoverable robust and minimizes

1. the nominal travel time $t_{nom}(P) = \pi_{\text{last}(P)} - s$ where $\text{last}(P)$ is the last arrival node on P *(the nominal objective function)*
2. the worst-case travel time $t_{wc}(P) = \max_{\xi \in \mathcal{U}} \min_{Q \in \mathcal{Q}^\xi(P)} \pi^\xi_{\text{last}(Q)} - s$ where $\text{last}(Q)$ is the last arrival event on Q *(the worst-case objective function).*

Note that for simplicity, we call $t_{wc}(P)$ the worst-case travel time of P, although the path P is only taken in the nominal case and an alternative path $P^\xi := \text{argmin}_{Q \in \mathcal{Q}^\xi(P)} \pi^\xi_{\text{last}(Q)} - s$ is taken in case of delay scenario ξ.

In other words, the bicriteria recoverable robust shortest path problem aims at finding paths which, on the one hand, are good in the nominal case, i. e., if no delays occur, and

on the other hand hedge against the scenarios from the uncertainty set $\mathcal{U}$ by minimizing worst-case travel time on the corresponding recovery paths.

3 Algorithms for Recoverable Robust Paths

3.1 A Recovery-Label Setting Algorithm

In this section we show that in case of finite uncertainty sets solutions to the bicriteria recoverable robust path problem can be found as solutions to a bicriteria minimax bottleneck shortest path problem in the EAN with *recovery labels* $L(a) := (L_{nom}(a), L_{wc}(a))^T$ at all arcs $a \in \bigcup_{\xi \in \mathcal{U}} A^\xi$. The *minimax bottleneck shortest path problem* is the problem of finding a path between two nodes in a network which minimizes $\max_{a \in P} c(a)$ in a graph with edge labels $c(a)$. In the bicriteria version of this problem, every arc a is assigned two different labels $c_1(a)$ and $c_2(a)$. We now state the preprocessing Algorithm 1 which calculates the recovery labels $L(a) := (L_{nom}(a), L_{wc}(a))^T$ needed to apply solution methods for the bicriteria minimax bottleneck shortest path problem. In Algorithm 1, for every $a = (i, j) \in \mathcal{A}^\xi$, $L^\xi(a)$ denotes the minimal travel time on a path which uses node j in scenario ξ. If no such path exists in scenario ξ, $L^\xi(a)$ is set to ∞. The algorithm returns the labels $L = (L_{nom}, L_{wc})^T$ which are 0 for all $a \notin \bigcup_{\xi \in \mathcal{U}} A^\xi$. For $a \in \bigcup_{\xi \in \mathcal{U}} A^\xi$, $L_{nom}(a)$ denotes the minimum nominal travel time when using a path containing node j, (and is ∞, if no such path exists) while $L_{wc}(a)$ represents the worst-case travel time for scenarios revealed at node j.

After initialization of all required labels to the value 0 (lines 1-6), we compute the shortest path distance from every event to the destination in the nominal scenario (line 7). This can be done by a single invocation of a standard shortest path tree computation in the reversed digraph from the destination v. Then, in the for-loop of lines 8-15, we iterate over all delay scenarios. With respect to the revealing time of scenario ξ, we now determine the set V^ξ. Using again a backward shortest path tree computation with respect to $\mathcal{N}^\xi$, we determine for every event $j \in \mathcal{E}$ the length of a shortest path towards the destination v. Using these values, we can set the nominal and worst-case labels for paths which go through arcs in $\mathcal{A}^\xi$ (lines 11-13). For ease of notation, we use $\infty + k = \infty$ for all values k. Note that the label $L_{nom}(a)$ is only set if the corresponding edge a can be used in some scenario $\xi \in \mathcal{U}$. We finally obtain the worst-case labels by taking the maximum over all scenarios. Note that lines 16-19 could be easily integrated into the main loop, but in the way presented here, the main loop can be run in parallel.

Given the recovery labels, the worst-case minimal travel time $t_{wc}(P)$ on a path P can be calculated as the maximum over the labels L_{wc} on P, as stated in the following lemma.

▶ **Lemma 3.** *Let P be a path from u to v in $\mathcal{N}$. Then for the labels calculated in Algorithm 1 it holds that*

- *if $\max_{a \in P} L_{wc}(a) < \infty$, P is recoverable robust, and*
- $t_{wc}(P) = \max_{a \in P} L_{wc}(a)$.

Proof. Consider an arbitrary scenario $\xi := (\mathcal{N}^\xi, \pi^\xi, U^\xi, V^\xi)$. The passenger travels on path P until node $j^\xi(P)$. Then, he/she can take the path calculated in step 10 of the algorithm until node v with total length $L^\xi(i^\xi(P), j^\xi(P))$ and this path has minimal length in $\mathcal{N}^\xi$ among all paths containing node j. We conclude that (1) P is recoverable robust, and (2) $t_{wc}(P) = \max_{a \in P} L_{wc}(a)$. ◀

For any path P, the labels L_{nom} constitute lower bounds on the nominal travel time on P. However, for an arbitrary path P, the nominal traveling time can exceed $\max_{a \in P} L_{nom}(a)$. This can be avoided for paths which do not make detours after the scenarios are revealed.

Algorithm 1 Construction of recovery labels

Require: EAN $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ with timetable π, origin node u, destination node v, starting time s, and finite uncertainty set $\mathcal{U}$.
Ensure: Label $L(a) \in \mathbb{R}^2_+$ for every $a \in \mathcal{A}$.

```
for (i,j) ∈ A do                                          ▷ Initialization
    Set L_nom(i,j) := 0.
    for ξ ∈ U do
        Set L^ξ(i,j) := 0.
    end for
end for
Find length K_nom(j) of shortest path from every j ∈ E to v in N. Set K_nom(j) := ∞ if
no such path exists.
for ξ ∈ U do
    Determine A^ξ.
    Find length K^ξ_wc(j) of shortest path from every j ∈ E to v in N^ξ. Set K^ξ_wc(j) := ∞
    if no such path exists.
    for (i,j) ∈ A^ξ do
        Set L_nom(i,j) := π_j − s + K_nom(j).             ▷ Setting nominal labels.
        Set L^ξ(i,j) := π^ξ_j − s + K^ξ_wc(j).            ▷ Setting worst-case labels.
    end for
end for
for (i,j) ∈ A do
    Set L_wc(i,j) := max_{ξ∈U} L^ξ(i,j)
    Set L(i,j) := (L_nom(i,j), L_wc(i,j))^T.
end for
return L
```

▶ **Lemma 4.** *Let P be a path from u to v in $\mathcal{N}$ such that the path P^2 defined as the subpath of path P starting in the last arc (i,j) in $P \cap \left(\bigcup_{\xi \in \mathcal{U}} \mathcal{A}^\xi\right)$ is a shortest path from j to v. Then for the labels calculated in Algorithm 1 it holds that*

- *if $\max_{a \in P} L_{wc}(a) < \infty$, P is recoverable robust,*
- *$t_{nom}(P) = \max_{a \in P} L_{nom}(a)$, and*
- *$t_{wc}(P) = \max_{a \in P} L_{wc}(a)$.*

Proof. This follows from Lemma 3 and from the construction of the labels L_{nom} in Algorithm 1 as the sum of the travel time $\pi_j - s$ until node j and the shortest path travel time $K_{nom}(j)$ from j to v. ◀

As a conclusion, we obtain the following theorem.

▶ **Theorem 5.** *The bicriteria recoverable robust path problem corresponds to a bicriteria bottleneck shortest path problem in the EAN with labels L.*

It is folklore that the single-criteria bottleneck shortest path problem can be solved in linear time on directed acyclic graphs. The Pareto front of bicriteria bottleneck shortest path problems can be found in $O(|\mathcal{A}|^2)$ by a simple ε-constraint method which enumerates all possible values of the first objective function, deletes edges whose labels exceed the given value, and finds a bottleneck shortest path with respect to the second criterion in the remaining graph (compare [12]).

▶ **Lemma 6.** *Algorithm 1 determines the labels L in time $O(|\mathcal{A}| \cdot |\mathcal{U}|)$.*

Proof. The initialization takes time $O(|\mathcal{A}| \cdot |\mathcal{U}|)$. Since we can assume that $\mathcal{N}$ is topologically sorted, shortest paths from a node to all other nodes can be found in time $O(|\mathcal{A}|)$. Hence, step 7 takes time $O(|\mathcal{A}|)$. For every $\xi \in \mathcal{U}$, determining $\mathcal{A}^\xi$ is in $O(|\mathcal{A}|)$. Since step 10 again is a shortest path calculation in a topologically sorted network and the operations in the loop over all $(i,j) \in \bigcup_{\xi \in \mathcal{U}} A^\xi$ take constant time, steps 8-15 can be executed in time $O(|\mathcal{A}| \cdot |\mathcal{U}|)$. ◀

3.2 Single-Criteria Versions of Recoverable Robustness

To calculate the Pareto front of the bicriteria recoverable robust path problem with finite uncertainty set, we can use the approach as sketched in the previous section. However, we are also interested in two single-criteria versions of the problem. In particular, results of versions with single objective values can be much easier compared for sets of instances.

▶ **Problem 2.** *Worst-case optimal recoverable robust paths*
Find a recoverable robust path P from u to v in $\mathcal{N}$ such that
- the nominal quality of P is smaller or equal than a given nominal quality bound T_{nom},
- P minimizes $t_{wc}(P)$.

▶ **Problem 3.** *Nominally optimal recoverable robust paths*
Find a recoverable robust path P from u to v in $\mathcal{N}$ such that
- the worst-case quality of P is smaller or equal than a given worst-case quality bound T_{wc}
- P minimizes $t_{nom}(P)$.

Algorithm 2 describes how to compute worst-case optimal recoverable robust paths. The pseudo-code for an analogous algorithm to compute nominally optimal recoverable robust path, Algorithm 3, is provided in the Appendix.

Algorithm 2 Worst-case optimal recoverable robust path

Require: Network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$, labels L, nominal quality bound T_{nom}, origin event u, destination event v.
Ensure: Path P which is optimal for Problem 2 (if existing).

```
1:  for a ∈ A do
2:      if L_nom(a) > T_nom then
3:          Remove a from A.
4:      end if
5:  end for
6:  Find a bottleneck shortest path P_wc in N according to labels L_wc.
7:  if there is no such path with length < ∞ then
8:      return There is no recoverable robust path.
9:  else
10:     Let (i, j) be the last arc on P_wc ∩ ⋃_{ξ∈U} A^ξ.
11:     Denote by P^1(j) the path P_wc until node j.
12:     Find a shortest path P^2(j) in N from j to v.
13:     return P := P^1(j) ∪ P^2(j), t_nom(P) := max_{a∈P} L_nom(a), t_wc(P) := max_{a∈P} L_wc(a)

14: end if
```

Table 1 Characteristics of the used event activity network and test queries.

characteristic	event activity network
# trains	38,495
# events	2,015,664
# stations	8,857
# transfer activities	19,869,867
aver. nominal travel time	398 min
aver. # transfers per query	3.3

▶ **Lemma 7.** *Algorithm 2 and Algorithm 3 are correct.*

Proof. Let P be the path returned by Algorithm 2 or Algorithm 3. Then, due to the construction of P in step 13 of each algorithm, the assumptions of Lemma 4 are fulfilled, i.e.,

- since $\max_{a \in P} L_{wc}(a) < \infty$, P is recoverable robust,
- $t_{nom}(P) = \max_{a \in P} L_{nom}(a)$, and
- $t_{wc}(P) = \max_{a \in P} L_{wc}(a)$.

Since there is no arc a with $L_{nom} > T_{nom}$ (or $L_{wc} > T_{wc}$, respectively) we have that $t_{nom}(P) \leq T_{nom}$ (or ($t_{wc}(P) \leq T_{wc}$, respectively). Furthermore, for any other path P' we have that

$$t_{wc}(P') = \max_{a \in P'} L_{wc}(a) \geq \max_{a \in P} L_{wc}(a) = t_{wc}(P)$$

(or $t_{nom}(P') = \max_{a \in P'} L_{nom}(a) \geq \max_{a \in P} L_{nom}(a) = t_{nom}(P)$, respectively). ◀

4 Experimental Results

4.1 Test Instances

The basis for our computational study is the German train schedule of February 1, 2013 from which we created an event-activity network. We generated transfer activities between pairs of trains at the same station provided that the departing train is scheduled to depart not later than 60 minutes after the planned arrival time of the feeding train. In addition, since some train lines operate only every two hours or irregularly, we add further transfer arcs. Namely, for each arrival event at some station s, we also create a transfer arc to those departure events which exceed the time bound of 60 minutes but provide the very next opportunity to get to a neighboring station. The main characteristics of the resulting network are shown in Table 1. To study the robustness of passenger paths, queries should not be too easy. For example, we are not interested in paths which do not require any transfer. Therefore, we decided to generate 1000 relatively difficult queries as follows. For each query, origin and destination are chosen uniformly at random from a set of the 3549 most important stations in Germany (this choice of stations has been provided by Deutsche Bahn AG). Such a pair of origin and destination stations is only accepted if the air distance between them is at least 200km and if the shortest travel route between them requires at least one transfer. The desired start time is uniquely set to 8:00am. The resulting set of queries has an average nominal travel time of 398 minutes and 3.3 transfers per query.

4.2 Generating Information Scenarios

A *delay scenario* $d \in \mathbb{N}_0^{\mathcal{A}_{drive} \cup \mathcal{A}_{wait}}$ specifies a delay on each driving and waiting activity. To generate a delay scenario, we first choose the revealing time of the scenario. Afterwards, we decide for each driving and waiting activity whether it shall receive a source delay or not. We use a parameter $p \in (0, 1)$ specifying the probability that a train receives a source delay. This parameter p can be chosen depending on the level of robustness one wants to achieve.

If a train shall be source-delayed, we select one of its driving or waiting activities uniformly at random from those which are scheduled after the revealing time of the scenario and choose the source delay for this activity uniformly at random among 10, 15, 20, 25, and 30 minutes. The source delays on all other activities are set to 0. For simplicity, we assume that trains receive source delays independently from each other.

We use the following basic *delay propagation rule* in order to compute how delays spread along driving, waiting and maintained transfer activities: $\pi(d)$ denotes the timetable adapted to delay scenario d. If the start event of an activity $a = (i, j)$ is delayed, also its end event j will be delayed, where the delay can be reduced by the slack time b_a. I.e. we require $\pi(d) \geq \pi$ and

$$\pi_j(d) \geq \pi_i(d) + l_a + d_a \tag{1}$$

for all activities $a = (i, j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive}$. For transfer activities equation (1) does not necessarily hold. Motivated by real-world decision systems of rail operators, we assume that the decision whether a transfer is actively maintained or not is specified by a *fixed waiting time rule*: Given a number $wt_a \in \mathbb{N}$ for every transfer activity, the transfer is actively maintained if the departing train has to wait at most wt_a minutes compared to its original schedule. If transfer a is actively maintained, we require that (1) holds for it. However, if for a transfer activity $a = (i, j)$ (1) holds due to some earlier delay on the train corresponding to j, a is maintained, even if $\pi_j(d) - \pi_j > wt_a$. Hence, every delay d induces a new set of transfer activities which is denoted as $\mathcal{A}_{transfer}(d)$. Given these waiting time rules for a given delay scenario d we can propagate the delay through the network along the activities in $\mathcal{A}_{drive} \cup \mathcal{A}_{wait} \cup \mathcal{A}_{trans}(d)$ and, thus, calculate the corresponding adapted timetable according to the following propagation rule:

$$\pi_j(d) = \max\left\{\pi_j, \max_{i:(i,j)\in\mathcal{A};\ \pi_i(d)+l_{ij}\leq\pi_j+wt_{ij}} \{\pi_i(d) + l_{ij} + d_{ij}\}\right\} \tag{2}$$

where we set $wt_a = \infty\ \forall a \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive}$ and $d_a = 0\ \forall a \in \mathcal{A}_{trans}$. The concrete waiting time rule used in our experiments is that high speed trains (like Intercity Express ICE, Intercity IC, and Eurocity EC) wait for each other at most three minutes, whereas trains of other train categories do not wait. Note that delay propagation can be done in time $O(|\mathcal{A}|)$. The uncertainty sets used in our experiments contain a number k of independent scenarios generated as described above.

4.3 Environment

All experiments were run on a PC (Intel(R) Xeon(R), 2.93GHz, 4MB cache, 47GB main memory under Ubuntu Linux version 12.04 LTS). Only one core has been used by our program. Our code is written in C++ and has been compiled with g++ 4.6.3 and compile option -O3.

4.4 Experiments

The purpose of this study is to evaluate the potential of recoverable robust paths as an alternative timetable information method in pretrip planning. A standard way of doing timetable information is to search for a path with minimum travel time as primary objective and with minimum transfers as a secondary one. We take this kind of standard search as the baseline of our comparisons.

Experiment 1: What is the effect of delays on the paths of the standard search? We perform the following evaluation. Suppose that P is a given path. For each delay scenario, we determine the first event after the scenario's revealing time. We assume that the passenger can adjust his/her path to the delay scenario at this point and therefore compute the earliest arrival time at the destination under these conditions. The worst-case arrival time over all scenarios is the value we are interested in. To each of our 1000 test queries we applied the same set of 100 delay scenarios with parameter $p = 0.20$. We observe that on average the worst-case travel time is 450 minutes, i.e., 13% larger than the planned one. The absolute difference is 52 minutes on average.

Experiment 2: What is the price of a worst-case optimal recoverable robust path in comparison with a standard path? Using the same 100 delay scenarios as for Experiment 1, we are interested in two quantities, namely the nominal travel time and the worst-case travel time of a worst-case recoverable robust path. We upper bounded the nominal arrival time of a recoverable robust path by 150% of the fastest nominal path. Among all paths satisfying this bound we minimized the worst-case arrival time over all scenarios. Our computational results show that for all 1000 queries but two cases there exists a recoverable robust path. An interesting observation is that 34.2% of all standard paths are already the worst-case optimal recoverable robust paths. However, in 27% of the queries the worst-case arrival time is improved in comparison with the standard path. If there is an improvement, the reduction is 29 minutes on average, but the maximum observed difference is 220 minutes. The histogram in Figure 1 gives a more detailed picture. It shows how often a saving of x minutes in the worst-case scenario can be achieved by choosing a recoverable robust path. The price a passenger has to pay if he/she chooses a recoverable robust path is a slight average increase in nominal travel time to 407 minutes, i.e., about just 9 minutes more than for the standard search.

In Figure 2, we show box-and-whisker plots for the distributions of travel times for five algorithmic variants. The data is based on our test set of 1000 queries, each evaluated for 100 delay scenarios generated with parameter $p = 0.2$ for the probability that a train will be delayed by a source delay.

Recall that StNom and StWC stand for the nominal and worst-case travel time in minutes of the standard search, while RRNom and RRWC denote the nominal and worst-case travel time for worst-case optimal recoverable robust paths, respectively. Finally, SRNom gives the nominal travel time for strictly robust paths.

Experiment 3: What is the influence of parameter p, initially chosen as $p = 0.2$? Recall that parameter p specifies the probability that a train will be delayed by a source delay. To quantify the sensitivity of the different solution methods on the chosen uncertainty set, we redo the previous two experiments with $p = 0.1$ and $p = 0.15$. Figure 3 (left) and Table 2 summarize our findings and show the average nominal (StNom) and worst case travel time (StWC) in minutes for the standard search and the nominal (RRNom) and worst-case time (RRWC) for the optimal recoverable robust paths, respectively. If the probability parameter p increases, we observe a slight increase of average worst case travel times (what

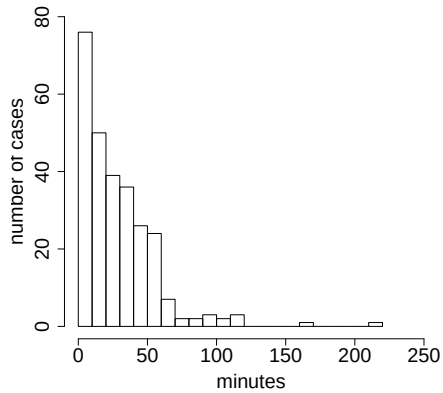

Figure 1 This histogram shows the number of cases where with respect to the worst-case scenario we can save x minutes by choosing a worst-case optimal recoverable robust path instead of the standard path.

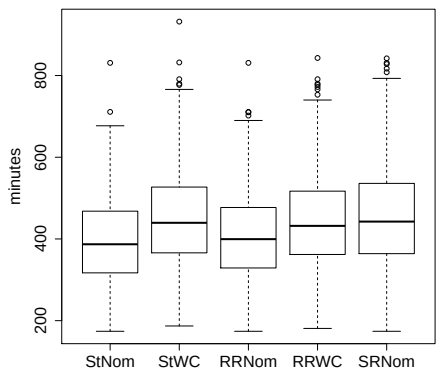

Figure 2 Box-and-whisker plots for the travel time distributions of several algorithmic variants.

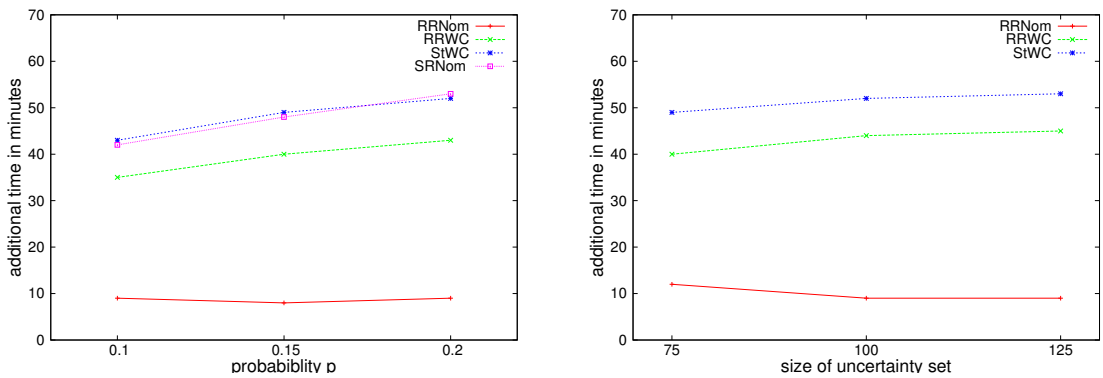

Figure 3 Additional travel time over the baseline of the standard path in minutes for different values of probability paramter p (left) and different number of scenarios (right). The average nominal travel time for standard paths is 498 minutes.

should be expected), whereas the nominal travel time of recoverable robust paths is almost unchanged. We conclude that $p = 0.2$ might be preferable since it provides recoverability for the more severe scenarios at no price with respect to nominal travel time.

Table 2 shows the raw data from which Figure 3 (left) has been derived.

Experiment 4: Comparison with strictly robust paths. Using the same uncertainty set as in the previous experiments, we computed the set of transfer activities which break at least once. We marked these arcs as forbidden, and rerun shortest path queries on the resulting even-activity network. Paths in this network are considered as strictly robust since no transfer will ever break. The average nominal travel time if we look for the fastest strictly robust path (SRNom) is 451 minutes for the uncertainty set with $p = 0.2$ (see also Figure 3 (left) and the last row of Table 2. Hence, the average nominal travel time of these paths is not better than the average worst-case time for standard paths. In full agreement with previous studies [17, 18], strictly robust paths turn out to be too conservative.

Table 2 Comparison of standard and robust solutions: Average travel time in minutes for $k = |U| = 100$ scenarios.

	$p = 0.10$	$p = 0.15$	$p = 0.20$
StNom	398	398	398
StWC	441	447	450
RRNom	407	406	407
RRWC	433	438	442
SRNom	440	446	451

Table 3 Comparison of standard and robust solutions for different sizes k of the uncertainty set: Average travel time in minutes for $p = 0.20$.

	$k = 75$	$k = 100$	$k = 125$
StNom	398	398	398
StWC	447	450	451
RRNom	410	407	407
RRWC	438	442	443

Experiment 5: To which extent do our observations depend on the size of the scenario set? All previous experiments have been run with 100 different delay scenarios. The parameter $k = |\mathcal{U}|$ has been chosen as a pragmatic compromise between efficiency (the computational effort scales linearly with k) and the degree of robustness we want to guarantee. Obviously, the more different scenarios we use, the higher the level of robustness we can achieve. Therefore, we fixed the parameter $p = 0.20$ but varied $k \in \{75, 100, 125\}$. Table 3 shows the average travel times in minutes for these variants, and Figure 3 (right) displays the additional travel time over the baseline of the standard path in minutes. It is interesting to observe that the average worst-case travel times depend only marginally on the parameter k in the chosen range. As expected, there is a slight increase of a few minutes on worst-case travel time when we increase k. At the same time, the average nominal travel time for recoverable robust paths does not increase. Further experiments will be needed to see whether this trend will be confirmed if k is chosen in an even wider range.

Practicality of our approach. For the purpose of this study, we have merely implemented a first prototype without much emphasis on performance issues. Our running times are several minutes per query which is clearly impractical. The main bottleneck is the computation of labels which grows linearly with the number of used scenarios. However, the most expensive part, namely the loop of lines 8-15, could be run in parallel. Thus, using massive parallelization and further speed-up techniques, we see a clear perspective that the computation time for a recoverable robust path can be brought down to a few seconds.

5 Conclusion and Further Research

In this work we introduced the concept of time-dependent recoverable-robust paths within the framework of timetable information. We showed that the resulting bicriteria problem can be solved in polynomial time using a label-setting algorithm, and a subsequent bottleneck shortest path calculation. The proposed concept and algorithm was experimentally evaluated on timetable information instances covering the whole German train network (schedule of 2013). While computation times are still too high for practical applications in the current implementation, we may assume that a parallelized algorithm will be sufficiently fast; moreover, as our experiments show that the proposed model has a valuable trade-off between nominal and worst-case travel times, such an algorithm will provide a customer-friendly alternative in practice. Further research includes the comparison of recoverable robust paths to lightly robust paths (see [17, 18]), and the extension of the proposed model to multi-stage robustness where only partial information on the scenario is given at discrete points in time. Also, the evaluation of the computed paths with respect to a set of real delay scenarios is currently being analyzed.

References

1 A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Math. Programming A*, 99:351–376, 2003.

2 A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

3 A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Math. Programming A*, 88:411–424, 2000.

4 A. Berger, M. Grimmer, and M. Müller-Hannemann. Fully dynamic speed-up techniques for multi-criteria shortest paths searches in time-dependent networks. In P. Festa, editor, *Proceedings of SEA 2010*, volume 6049 of *LNCS*, pages 35–46. Springer, Heidelberg, 2010.

5 D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

6 C. Büsing. Recoverable robust shortest path problems. *Networks*, 59(1):181–189, 2012.

7 C. Büsing, A. M. C. A. Koster, and M. Kutschka. Recoverable robust knapsacks: the discrete scenario case. *Optimization Letters*, 5(3):379–392, 2011.

8 V. Cacchiani, A. Caprara, L. Galli, L. Kroon, G. Maroti, and P. Toth. Railway rolling stock planning: Robustness against large disruptions. *Transportation Science*, 46(2):217–232, May 2012.

9 A. Caprara, L. Galli, S. Stiller, and P. Toth. Recoverable-robust platforming by network buffering. Technical Report ARRIVAL-TR-0157, ARRIVAL Project, 2008.

10 S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, and A. Navarra. Robust Algorithms and Price of Robustness in Shunting Problems. In *ATMOS 2007*, 2007.

11 S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, A. Navarra, M. Schachtebeck, and A. Schöbel. Recoverable robustness in shunting and timetabling. In *Robust and Online Large-Scale Optimization*, volume 5868 of *LNCS*, pages 28–60. Springer, Heidelberg, 2009.

12 L. de Lima Pinto, C. T. Bornstein, and N. Maculan. The tricriterion shortest path problem with at least two bottleneck objective functions. *European Journal of Operational Research*, 198:387–391, 2009.

13 J. Dibbelt, Th. Pajor, B. Strasser, and D. Wagner. Intriguingly simple and fast transit routing. In V. Bonifaci, C. Demetrescu, and A. Marchetti-Spaccamela, editors, *Experimental Algorithms*, volume 7933 of *LNCS*, pages 43–54. Springer, Heidelberg, 2013.

14 Y. Disser, M. Müller-Hannemann, and M. Schnee. Multi-criteria shortest paths in time-dependent train networks. In C. C. McGeoch, editor, *WEA 2008*, volume 5038 of *LNCS*, pages 347–361. Springer, Heidelberg, 2008.

15 A.L. Erera, J.C. Morales, and M. Savelsbergh. Robust optimization for empty repositioning problems. *Operations Research*, 57(2):468–483, 2009.

16 M. Fischetti and M. Monaci. Light robustness. In R. K. Ahuja, R.H. Möhring, and C.D. Zaroliagis, editors, *Robust and online large-scale optimization*, volume 5868 of *LNCS*, pages 61–84. Springer, Heidelberg, 2009.

17 M. Goerigk, M. Knoth, M. Müller-Hannemann, M. Schmidt, and A. Schöbel. The price of robustness in timetable information. In *ATMOS 2011*, volume 20 of *OASICS*, pages 76–87. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2011.

18 M. Goerigk, M. Knoth, M. Schmidt, A. Schöbel, and M. Müller-Hannemann. The price of strict and light robustness in timetable information. *Transportation Science*, 2013. To appear.

19 C. Liebchen, M. Lübbecke, R. H. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In R. K. Ahuja, R.H. Möhring, and C.D. Zaroliagis, editors, *Robust and online large-scale optimization*, volume 5868 of *LNCS*, pages 1–27. Springer, Heidelberg, 2009.

20 M. Müller-Hannemann and M. Schnee. Efficient timetable information in the presence of delays. In R. Ahuja, R.-H. Möhring, and C. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *LNCS*, pages 249–272. Springer, Heidelberg, 2009.

21 M. Müller-Hannemann, F. Schulz, D. Wagner, and C. Zaroliagis. Timetable information: Models and algorithms. In *Algorithmic Methods for Railway Optimization*, volume 4395 of *LNCS*, pages 67–89. Springer, Heidelberg, 2007.

22 M. Schnee. *Fully realistic multi-criteria timetable information systems*. PhD thesis, Fachbereich Informatik, Technische Universität Darmstadt, 2009. Published in 2010 by Südwestdeutscher Verlag für Hochschulschriften.

23 O. Seref, A. Ravindra, and J. B. Orlin. Incremental network optimization: Theory and algorithms. *Operations Research*, 57(3):586–594, 2009.

24 A.L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21:1154–1157, 1973.

25 S. Stiller. *Extending concepts of reliability. Network creation games, real-time scheduling, and robust optimization*. PhD thesis, TU Berlin, 2008.

A Algorithmic Approach

Algorithm 3 describes how to find nominally optimal recoverable robust paths subject to an upper worst-case quality bound.

Algorithm 3 Nominally optimal recoverable robust path

Require: Network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$, labels L, worst-case quality bound T_{wc}, origin event u, destination event v.

Ensure: Path P which is optimal for Problem 3 (if existing).

```
for a ∈ A do
    if L_wc(a) > T_wc then
        Remove a from A.
    end if
end for
Find a bottleneck shortest path P_nom in N according to labels L_nom.
if there is no such path with length < ∞ then
    return There is no recoverable robust path.
else
    Let (i, j) be the last arc on P_nom ∩ ⋃_{ξ∈U} A^ξ.
    Denote by P^1(j) the path P_nom until node j.
    Find a shortest path P^2(j) in N from j to v.
    return P := P^1(j) ∪ P^2(j), t_nom(P) := max_{a∈P} L_nom(a), t_wc(P) := max_{a∈P} L_wc(a)
end if
```