

Robust Algorithms for Sorting Railway Cars

Christina Büsing¹ and Jens Maue²

¹ Institut für Mathematik, Technische Universität Berlin, Germany
`cbuesing@math.tu-berlin.de`

² Institute of Theoretical Computer Science, ETH Zürich, Switzerland
`jens.maue@inf.ethz.ch`

Abstract. We consider a sorting problem from railway optimization called train classification: incoming trains are split up into their single cars and reassembled to form new outgoing trains. Trains are subject to delay, which may turn a prepared sorting schedule infeasible for the disturbed situation. The classification methods applied today deal with this issue by completely disregarding the input order of cars, which provides robustness against any amount of disturbance but also wastes the potential contained in the a priori knowledge about the input.

We introduce a new method that provides a feasible sorting schedule for the expected input and allows to flexibly insert additional sorting steps if the schedule has become infeasible after revealing the disturbed input. By excluding disruptions that almost never occur from our consideration, we obtain a classification process that is quicker than the current railway practice but still provides robustness against realistic delays. In fact, our algorithm allows flexibly trading off fast classification against high degrees of robustness depending on the respective need. We further explore this flexibility in experiments on real-world traffic data, underlining our algorithm improves on the methods currently applied in practice.

1 Introduction

An essential process in railway optimization is *train classification*, which refers to the rearrangement of cars to form new trains. With increasing world-wide freight traffic, operating freight trains efficiently becomes more and more important, and reducing the dwell time of cars in railway yards is one of the key factors to improve freight service profitability.

General Classification Process. Exclusively for the purpose of train classification, there are installations of railway tracks and switches called *classification yards* (see Fig. 1). Such a yard features a *hump track* on which inbound trains arrive and their cars are decoupled to be pushed over a sloping ramp called *hump* at the end of the hump track. Hence, the cars accelerate by gravity and roll through a tree of switches by which each car can be individually guided to some *classification track*. This is called a *roll-in* operation. In a *pull-out* operation an engine pulls all the cars on some classification track back to the hump track in order to perform a further roll-in. A pair of pull-out and roll-in operations is

called a (*sorting*) *step*, and an initial roll-in followed by a sequence of h sorting steps is called a *classification schedule* of *length* h . The number of steps h essentially determines the time required to conclude the sorting procedure. There are ℓ inbound trains that, concatenated in the order they arrive at the yard, form the *inbound train sequence*. Moreover, there are order specifications for the m *outbound trains*, and a classification schedule is called *feasible* if its application to the inbound train sequence yields the correctly ordered outbound trains, each on a separate classification track.

Robust Train Classification. Often the inbound trains are subject to delay, so we might be faced with an unexpected inbound order of trains. In our model every possible number of delayed trains and the amount of delay for each train are given by a *set of scenarios* \mathcal{S} . Each scenario $S \in \mathcal{S}$ defines a *modified instance*, which is a permutation of the original inbound train sequence. A schedule for the original instance is called a *first-stage decision*, and this schedule may be infeasible for the modified instance corresponding to some scenario. In response to disturbed input, we are prepared to insert up to k additional sorting steps after the p th step of the first-stage decision, providing a *recovered solution*. A first-stage decision for which, for every scenario $S \in \mathcal{S}$, there is a recovered solution that is feasible w.r.t. S is called *recovery robust*. Given a sequence of ℓ inbound trains, m order specifications of outbound trains, and a set of scenarios \mathcal{S} , the recovery-robust train classification problem is to find a recovery robust, feasible first-stage decision of minimum length.

Related Work. There are many publications in the field of railway engineering that describe different train classification methods, e.g. [7, 17, 14, 18, 5]. These methods are *strictly robust*, i.e. robust w.r.t. any set of scenarios, since they apply a predefined classification schedule that is independent of the order of railway cars entering the classification process. The method of *geometric sorting* (see [7, 17, 14, 18]) minimizes the number of sorting steps for a worst case (or unknown) input order, which is proved in [12]. The still most-commonly used method in practice is *triangular sorting* [7, 17, 14, 18, 5], which is optimal for restricting the number of roll-ins per car to three for unknown input order [12]. However, neither method exploits the situation of a partially ordered input sequence, so they apply more sorting steps than necessary in general.

This issue was explored in [12], which develops a classification method that minimizes the number of sorting steps based on complete knowledge of the input data. Moreover, for the problem variant of classification tracks of bounded length, [12] shows that minimizing the number of sorting steps is an NP-hard problem. A 2-approximation for the same setting is derived in [13], several improvements of which are experimentally evaluated in [11] and compared to an exact integer programming approach, which was earlier introduced in [16]. A related algorithmic sorting problem is considered by Dahlhaus et al. [6]. Recent overviews of train classification can be found in [10] and [9].

Since changes during the process of scheduling are time consuming, a certain amount of robustness is crucial for classification methods to work in practice. Providing strict robustness, however, wastes a lot of potential to disruption sce-

narios that almost never occur in practice. As described above w.r.t. train classification, this dilemma is tackled by the concept of recoverable robustness [15] by regarding realistic scenarios of delay and providing optimal robust solutions w.r.t. a limited amount of recovery in case of disturbance. This concept is applied to several railway-related optimization problems such as rolling stock scheduling [1] or timetabling [3, 4]. A first and—to the best of our knowledge—only attempt to study this method for train classification is made by Cicerone et al [2] for a single inbound and outbound train. (Their results are summarized in [3].) Besides the situations of strict robustness and complete recomputation from scratch, which are more of theoretical interest, they consider a recovery action that allows completely changing the classification instruction for one set of cars that have the same instruction. The most relevant scenarios in [2] are one additional car in the input and one car occurring at a different position than expected. The latter corresponds to our problem setting for the special case of trains consisting of single cars with a delay scenario of up to one train. We generalize this setting to scenarios with more delays (mainly Sect. 4) and the problem setting with complex trains. Besides, [2] deals with the scenario of a single classification track becoming unavailable before the classification starts. In this paper we focus on the most relevant reason for disruptions, which are delayed trains.

Our Contribution. For the mentioned recovery action of adding up to k sorting steps after an offset of p steps, we first introduce a generic algorithm in Sect. 3. We prove that, for every constant $k \geq 1$, finding a robust schedule of minimum length is an **NP**-complete problem for general sets of scenarios. For the practically relevant scenario of delaying up to j trains by an arbitrary amount each, the problem can be solved in polynomial time (see Sect. 4). Furthermore, we evaluate our new algorithm on real-world traffic data for various parameter values k , p , and j . It turns out that, on the one hand, our algorithm yields very short schedules while providing a fair degree of robustness. On the other hand, it is capable of providing highly robust schedules that still improve on the current classification practice, emphasizing the flexibility of our approach to modulate between these conflicting objectives.

2 Encoding Classification Schedules

In addition to the concepts of Sect. 1, we introduce some further notation required for representing and deriving classification schedules.

Terminology and Notation. Corresponding to the notation of [12], we represent every car τ by a positive integer $\tau \in \mathbb{N}$ and a train T by a sequence of cars $T = (\tau_1, \dots, \tau_k)$, where k is called the *length* of T . There are ℓ *inbound* trains T_1, \dots, T_ℓ , whose concatenation we assume to be a permutation of $(1, \dots, n)$, and n is called the *volume* of cars. There are m *outbound* trains of respective length n_i , $i = 1, \dots, m$, and we assume the specification of the first outbound train is $(1, \dots, n_1)$, the second (n_1+1, \dots, n_1+n_2) , etc. In contrast to the expected order of inbound trains, there is no order implied for the outbound trains.

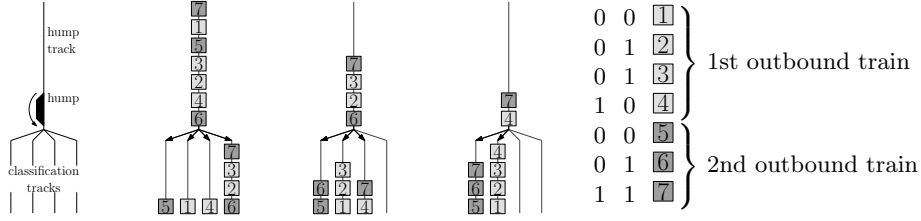


Fig. 1. Classification yard (left). Classification process for $h=2$, $n=7$, and $m=2$: initial roll-in (2nd picture), first step (3rd), second step (4th); in both steps, the rightmost occupied track is pulled out. Corresponding schedule encoding (right).

Regardless of which are their outbound trains, all cars are sorted simultaneously on the same set of classification tracks, called the *sorting tracks*. Their lengths and available number are unrestricted, and the number actually used corresponds to the number of sorting steps, where the track pulled in the k th step is referred to by θ_k , $k = 0, \dots, h-1$. The cars are finally collected on a separate track for each outbound train, which are called *destination tracks*.

Schedule Representation. We will refer to the binary representation of a decimal integer $j \geq 0$ by $[j]_2$. Given any bitstring $b = b_{h-1} \dots b_0$ of length h , let $\text{num}(b)$ denote the integer number represented by b , i.e., $\text{num}(b) = \sum_{i=0}^{h-1} 2^i b_i$. For two bitstrings b_1, b_2 we define $b_1 < b_2$ iff $\text{num}(b_1) < \text{num}(b_2)$. We represent classification schedules of length h by assignments of cars to bitstrings of length h [12]: $b^j = b_{h-1}^j \dots b_0^j$ encodes the journey of the j th car with $b_k^j = 1$ iff it visits θ_k pulled out in the k th step. After such a pull-out, the car is sent to θ_ℓ with $\ell = \min\{i | k < i < h, b_i^j = 1\}$; if $b_i^j = 0$ for all $i > k$, it goes to the destination track of its outbound train. The n bitstrings b^1, \dots, b^n form an $(n \times h)$ -matrix, and b_0, \dots, b_{h-1} denote its columns (from “right” to “left”).

In order to derive a feasible schedule B of length h , two cars τ and $\tau+1$ of the same outgoing train must be assigned bitstrings $b^\tau \leq b^{\tau+1}$. If these cars occur in reversed order in the inbound sequence, we require $b^\tau < b^{\tau+1}$; then, the pair $\beta = (\tau, \tau+1)$ is called a *break*. If b^τ and $b^{\tau+1}$ occur in different outbound trains, there is no constraint between the two cars [13]. As a result the length of a classification schedule depends on the number of breaks. Figure 1 shows an example classification process and the corresponding encoding: cars 1 and 2 arrive in reversed order, so $b_1 < b_2$, whereas cars 2 and 3 arrive in correct order and have the same bitstring. Note that $b^7 > b^6$ is fine though cars 6 and 7 arrive in correct order, and there is no constraint between b_4 and b_5 since the fourth and fifth car belong to different outbound trains.

3 Recovery through Additional Sorting Steps

In this section we investigate the recovery strategy of inserting a limited number of additional sorting steps to a first-stage decision schedule when a scenario occurs. The proofs of this section can be found in App. A.

Further Notation. A pair of consecutive cars $\beta = (\tau, \tau+1)$ is called *original break* if β is a break for the expected order of inbound trains. Given some $S \in \mathcal{S}$, we call β *induced by S* if β is a break in the modified instance corresponding to S . If β is not an original break but induced by any $S \in \mathcal{S}$, β is called a *potential break*. W.l.o.g., we assume that every pair $\beta = (\tau, \tau+1)$, $\tau \in \{1, \dots, n-1\}$, of successive cars is either an original or a potential break: for any problem instance with β not being a break, car $\tau+1$ can be ignored while deriving a schedule and assigned the same bitstring as τ in the final solution. For any first-stage decision B , a break $\beta = (\tau, \tau+1)$ is called *unresolved* w.r.t. S if β is induced by S and $b^\tau = b^{\tau+1}$. For any scenario $S \in \mathcal{S}$, X^S denotes the set of potential breaks induced by S . Note that this set X^S is uniquely defined for every scenario $S \in \mathcal{S}$, but there may be different scenarios $S \neq S'$ with $X^S = X^{S'}$. We will repeatedly regard sets of potential breaks without considering the actual underlying scenario. In particular, we will often describe sets of scenarios (e.g. as a parameter in problem definitions) implicitly by providing the set of induced breaks of every scenario. Let T_1, \dots, T_ℓ be a sequence of ℓ inbound trains with n cars, and let X be the set of all original and potential breaks. For any pair of cars τ_1, τ_2 , $1 \leq \tau_1 < \tau_2 \leq n$, we define $X_{(\tau_1, \tau_2)}$ as the set of all original and potential breaks occurring between τ_1 and τ_2 , i.e., $X_{(\tau_1, \tau_2)} = X \cap \{(\tau_1, \tau_1+1), (\tau_1+1, \tau_1+2), \dots, (\tau_2-1, \tau_2)\}$.

Recovery Model. After a scenario is revealed, the original schedule may be infeasible. With the recovery action of inserting up to k additional sorting steps to the first stage decision, we seek to obtain a feasible schedule for the modified instance. The classification process according to the originally planned schedule may have started when the delay of a train is realized. Furthermore, distributing the recovered solution, i.e. the changed schedule, to all people involved in the operation takes some time depending on the available communication channels. For these reasons, inserting additional sorting steps is only allowed after an offset of p steps.

In terms of classification schedules, which present solutions to our optimization problem, this means the following: given two parameters $p \geq 0$ and $k \geq 0$ and a first-stage decision schedule B of length h , B is to be recovered by inserting up to k additional columns with indices greater than p . This concept is formalized in the following definition.

Definition 1. Let $B = (b_{h-1}, \dots, b_0)$ and $B' = (b'_{h-1+j}, \dots, b'_h)$ be two classification schedules for n cars of length h and $h+j$, $j \geq 0$, respectively. Let further $p \geq 0$ and $k \geq 0$. The schedule B' is called a (p, k) -extension of B if $j \leq k$, $b_i = b'_i$ for all $0 \leq i \leq p$ and $b_{i-p-j} = b'_i$ for all $p+j \leq i \leq h+j$.

Note that in this definition the additional columns are all added between the $(p-1)$ th and p th step of the original schedule since this always is the most powerful recovery. The notion of (p, k) -extensions yields a natural concept of recoverable robustness as stated in the following definition.

Definition 2. Let T_1, \dots, T_ℓ be a sequence of ℓ inbound trains and \mathcal{S} a set of scenarios with X^S denoting the corresponding induced set of breaks for every $S \in \mathcal{S}$. A classification schedule B is called (p, k) -recovery robust if, for every scenario $S \in \mathcal{S}$, there is a (p, k) -extension of B that is feasible w.r.t. S .

Most likely, no delay occurs and the inbound trains arrive in the expected order, in which case we usually do not want to apply any recovery for organizational reasons. For our objective this means we look for *feasible* (p, k) -recovery robust classification schedules of minimum length.

In order to specify when a given schedule is (p, k) -recovery robust for a given set of scenarios, we introduce the notion of a *block* of a schedule. A block basically is a maximal set of bitstrings representing integers between two powers of two.

Definition 3. Let B be a schedule of length h for an inbound train sequence of n cars, and $p \geq 1$. For any bitstring b^j of B , $b_{h-1}^j \dots b_p^j$ is called the leading part of b^j , denoted by $b_{>p}^j$, and $b_{p-1}^j \dots b_0^j$ the trailing part of B , denoted by $b_{<p}^j$. A subset of λ consecutive bitstrings $b^j, \dots, b^{j+\lambda-1}$ of B is called a block of B if their leading parts satisfy $b_{>p}^{j-1} < b_{>p}^j$, $b_{>p}^j = b_{>p}^{j+\lambda-1}$, and $b_{>p}^{j+\lambda-1} < b_{>p}^{j+\lambda}$, while λ is called the size of the block. Furthermore, the j th car of the inbound train sequence is called the head of the block.

The following lemma states the necessary and sufficient conditions for the existence of (p, k) -extensions. The second part of its proof (see App. A) presents a method to derive a (p, k) -extension for a given scenario S making use of the block structure of the original schedule. We will refer to this construction by *canonical* recovery and call the resulting schedule a *canonical* (p, k) -extension.

Lemma 1. Let T_1, \dots, T_ℓ be a sequence of ℓ inbound trains, B a feasible classification schedule, S a scenario, and $p, k \geq 0$. Then, there exists a (p, k) -extension of B that is feasible for S iff the number of unresolved breaks w.r.t. S does not exceed $2^k - 1$ for any block of B .

General Algorithm. Applying the observations of the previous section, we introduce a generic algorithm for computing (p, k) -recovery robust train classification schedules. Basically, the algorithm successively grows the size of a block to its maximum size. The maximum size of a block is determined by two factors: First, a schedule B assigns at most 2^p different bitstrings to the trailing part of cars in the same block, i.e., at most $2^p - 1$ breaks can be resolved. Secondly, the number of unresolved breaks in a block is limited by $2^k - 1$ potential breaks induced by one scenario. We formalize the second condition in the following way.

Definition 4. Let T_1, \dots, T_ℓ be a sequence of ℓ inbound trains with a total of n cars and $k \geq 0$. Given a set of scenarios \mathcal{S} , a set of breaks $X' \subseteq X_{(\tau_1, \tau_2)}$ is called k -recoverable according to $X_{(\tau_1, \tau_2)}$ if $|X' \cap X^S| \leq 2^k - 1$ holds for all $S \in \mathcal{S}$.

Algorithm 1 determines the maximum size of a block by repeatedly solving the problem of finding a maximum k -recoverable break set and thus constructs an optimal (p, k) -recovery robust schedule.

Theorem 1. For any $p \geq 0$ and $k \geq 0$, Alg. 1 computes an optimal (p, k) -recovery robust train classification schedule.

In Alg. 1 the step of computing a maximum k -recoverable break set in line 6 is not specified. One way of solving this problem would be by integer programming.

Algorithm 1 k -recovery robust train classification

Data: number of cars n , set of original breaks X_{org} , set of scenarios \mathcal{S} , $k, p \geq 0$

Result: k -recovery robust classification schedule B

```
1 Put  $i = 0$ ,  $\tau_i = 1$ ,  $\tau_{\max} = 0$ ,  $X = X' = \emptyset$ 
2 while  $\tau_i \leq n$  do
3   while  $\tau_{\max} < \tau_i + 2^p + |X'|$  and  $\tau_i + 2^p + |X'| \leq n$  do
4     Set  $\tau_{\max} = \tau_i + 2^p + |X'|$ 
5     Set  $X = X_{(\tau_i, \tau_{\max})} \cap (\cup_{S \in \mathcal{S}} X^S)$ 
6     Compute a maximum  $k$ -recoverable set of breaks  $X' \subseteq X$ 
7   end
8   Set  $\tau_{\max} = \tau_{i+1} = \min(\tau_i + 2^p + |X'|, n+1)$ 
9   Compute subschedule of length  $p$  for  $\tau_i, \dots, \tau_{i+1} - 1$  feasible w.r.t.  $X_{(\tau_i, \tau_{i+1}-1)} \setminus X'$ 
10  Set  $i = i+1$ 
11 end
12 Set  $h' = \lceil \log_2 i - 1 \rceil$ 
13 for  $j = 0, \dots, i-1$  do
14   Set  $b_{p+h'-1}^\tau \dots b_p^\tau = [j]_2$  for all  $\tau_j \leq \tau \leq \tau_{j+1} - 1$ 
15 end
16 return  $B$ 
```

As we will show in the following, there is in general no polynomial time algorithm to solve this problem unless $\mathbf{P} = \mathbf{NP}$.

Computational Complexity. In this section we assume w.l.o.g. that we are looking for a maximum k -recoverable break set for the cars $1, \dots, n$, i.e., let \mathcal{S} be a set of scenarios, find a maximum k -recoverable break set X' of $X = \cup_{S \in \mathcal{S}} X^S$. By a reduction from the independent set problem, the decision version of this problem is strongly \mathbf{NP} -hard for $k = 1$. A different reduction from 2^kSAT leads to the \mathbf{NP} -completeness for any constant $k \geq 2$.

Theorem 2. *Let T_1, \dots, T_ℓ be a sequence of ℓ inbound trains, \mathcal{S} a set of scenarios, and $K \geq 0$. For any constant $k \geq 1$, it is strongly \mathbf{NP} -complete to decide whether there exists a k -recoverable break set of size K .*

This theorem not only states that Alg. 1 will only run in polynomial time if $\mathbf{P} = \mathbf{NP}$ but also enable us to prove the \mathbf{NP} -completeness of the (p, k) -recovery robust classification problem.

Corollary 1. *Let T_1, \dots, T_ℓ be a sequence of ℓ inbound trains, \mathcal{S} a set of scenarios, $h, p \geq 0$, and $k \geq 1$ const. Deciding whether there is a feasible (p, k) -recovery robust classification schedule of length at most h is an \mathbf{NP} -complete problem.*

Infeasible Initial Decisions In our model the first stage decision is a feasible classification schedule for the original order of trains. A special case of this setting is, to allow recovery even in case of no disturbance. In this case, the original breaks are modeled by a scenario S_{org} with $X^{S_{\text{org}}} = X_{\text{org}}$ and no original breaks are consider, i.e., one assumes that the cars arrive in perfect order.

4 Limited Number of Delayed Trains

As mentioned before, providing strict robustness wastes a lot of potential to extreme scenarios that rarely occur. For this reason we introduce a simple yet general class of scenarios in this section. The proofs of this section can be found in App. B.

Scenario Model. Given some parameter j , up to j trains are delayed each by an arbitrary amount: let $\Theta = T_1, \dots, T_\ell$ be an inbound train sequence and $\Theta^\sigma = T_{\sigma^{-1}(1)}, \dots, T_{\sigma^{-1}(\ell)}$ be an order of trains induced by some permutation $\sigma: [\ell] \rightarrow [\ell]$. Then, a sequence $\bar{\Theta} = T_{\bar{\sigma}^{-1}(1)}, \dots, T_{\bar{\sigma}^{-1}(\ell)}$, where $\bar{\sigma}$ is some permutation, is called an (α, k) -delayed sequence of Θ^σ if $\sigma(\alpha) < k$ and the following conditions hold: $\bar{\sigma}(x) = \sigma(x)$ if $\sigma(x) < \sigma(\alpha)$ or $\sigma(x) > k$, $\bar{\sigma}(x) = \sigma(x) - 1$ for $\sigma(\alpha) < \sigma(x) < k$, and $\bar{\sigma}(\alpha) = k$. Less formally, train T_α is delayed from the $\sigma(\alpha)$ th to the k th position. The set of scenarios \mathcal{S}_j , $0 \leq j \leq \ell$, is now defined to contain a scenario S (inducing some sequence Θ^S) iff there is a sequence $\Theta^0, \dots, \Theta^j$ of train sequences Θ^i such that $\Theta^0 = \Theta$, Θ^i is an (α_i, k_i) -delayed sequence of Θ^{i-1} for all $i = 1, \dots, j$, and $\Theta^j = \Theta^S$. Every train T_{α_i} will furthermore be called to be *delayed* by S .

Dominating Set of Scenarios. We will see in Thrm. 3 that our considerations can be restricted to the dominating subset $\bar{\mathcal{S}}_j \subseteq \mathcal{S}_j$ of scenarios defined as follows: a scenario S is a member of $\bar{\mathcal{S}}_j$ iff there is a sequence $\Theta^0, \dots, \Theta^j$ of train sequences Θ^i such that $\Theta^0 = \Theta$, Θ^i is an (α_i, ℓ) -delayed sequence of Θ^{i-1} for all $i = 1, \dots, j$, $\alpha_i < \alpha_{i-1}$ for all $i = 1, \dots, j$, and $\Theta^j = \Theta^S$. In other words, if two trains are delayed by $S \in \bar{\mathcal{S}}_j$, they swap their relative order and arrive later than all punctual trains. Note that for uniquely defining a scenario $S \in \bar{\mathcal{S}}_j$ it suffices to list the j trains since the order and amount of their delay is determined by the definition of $\bar{\mathcal{S}}_j$.

Theorem 3. *Given any $p, k, j \geq 0$, let B be a feasible (p, k) -recovery robust schedule for $\bar{\mathcal{S}}_j$. Then, B is a feasible (p, k) -recovery robust schedule for \mathcal{S}_j .*

Any potential break $(\tau, \tau+1)$ can only be induced by S if the train containing τ is delayed, but also the converse implication holds for $\bar{\mathcal{S}}_j$ as shown in the following lemma.

Lemma 2. *Let T_1, \dots, T_ℓ be a sequence of inbound trains and $S \in \bar{\mathcal{S}}_j$ some scenario. For any potential break $\beta = (\tau, \tau+1)$ with $\tau \in T_x$, $x \in \{1, \dots, \ell\}$, $\beta \in X^S$ iff T_x is delayed by S .*

As an immediate consequence, the set of potential breaks X^S of any scenario $S \in \bar{\mathcal{S}}_j$ can be partitioned into disjoint subsets w.r.t. the respective delayed train causing the break, a fact which is applied in the algorithm of the following section. We will call the set $X_i := \{(\tau, \tau+1) | \tau \in T_i, \exists y > i : \tau+1 \in T_y\}$ the *set of breaks induced by train T_i* .

Maximum Recoverable Sets of Breaks. For $\bar{\mathcal{S}}_j$ a maximum recoverable set of breaks is computed with Alg. 2: we repeatedly resolve potential breaks of the train that induces the highest number of unresolved breaks until the worst case

Algorithm 2 Max. k -Recoverable Set of Breaks for \mathcal{S}_j with Unique Cars

Input: Parameters $j, k \in \mathbb{N}$ and sets of induced breaks X_1, \dots, X_ℓ **Output:** Maximum recoverable set of breaks

- 1 Descendingly sort X_1, \dots, X_ℓ such that $|X_{i_1}| \geq |X_{i_2}| \geq \dots \geq |X_{i_\ell}|$
 - 2 Put $\alpha := \max\{i_t : |X_{i_t}| = |X_{i_1}|\}$
 - 3 **while** $\sum_{t=1}^j |X_{i_t}| \geq 2^k$ **do**
 - 4 Remove an arbitrary break from X_{i_α}
 - 5 Put $\alpha := \max\{i_t : |X_{i_t}| = |X_{i_1}|\}$
 - 6 **end**
 - 7 **return** $\bigcup_{i=1}^\ell X_i$
-

scenario does not exceed the recovery capability given through the parameter k . Correctness, optimality, and the running time of Alg. 2 are summarized in the following theorem.

Theorem 4. *Given a set of potential breaks X for some classification instance with inbound trains T_1, \dots, T_ℓ , a maximum k -recoverable set of breaks $X' \subseteq X$ w.r.t. \bar{S}_j can be computed in polynomial time.*

As an immediate consequence of Thrm. 4, the problem of train classification can be solved in polynomial time by combining Alg. 2 into Alg. 1. The resulting algorithm is implemented in the following section and tested for a number of real-world classification instances.

Experimental Evaluation. For the evaluation of the algorithm just described, we took the five real-world instances used in [11], which correspond to five days of traffic in the Swiss classification yard Lausanne Triage. Their volumes range from 310 to 486 with numbers of inbound trains between 44 and 49, outbound trains between 24 and 27, and numbers of breaks between 24 and 28. (For details see App. C, Tab. 2.) In order to obtain unique types of cars, we converted all cars of the same type between two consecutive original breaks to distinct types ascending in the order the cars appear between the breaks. (Also see App. C.)

Essentially, through adjusting the parameters p , k , and j , the algorithm allows flexibly trading off shortest schedules against the other extreme of strict robustness. Given some train classification instance, let \underline{h} denote the length of an optimal non-robust schedule and \bar{h} the length of an optimal strictly robust schedule. The values \underline{h} and \bar{h} present the lower and upper bounds for the length resulting from any combination of j , k , and p . Yet, as explained in Sect. 1 (Related Work), \bar{h} may be exceeded by the geometric method, i.e. an optimal strictly robust schedule disregarding presorted inbound trains, and even longer schedules than this are obtained by triangular sorting.

Table 1 summarizes the computed length of an optimal recovery robust schedule according to the different parameters p , k and j . As lower and upper bounds for those length inst-2 requires $\underline{h} = 3$, while all other instances yield $\underline{h} = 2$ and $\bar{h} = 5$ for all instances. The geometric method requires $h = 6$ for three of the instances, and the triangular method even between eight (int-2 and inst-3) and

Table 1. Optimal length values for S_j with (p, k) -extensions for the five traffic instances: the values for the triangular and geometric method are given in the first and second column, resp., \bar{h} and \underline{h} in the third and fourth column, resp. Omitted entries represent no meaningful choice of p .

k	t	g	1										2										3										4									
			0										1										2										3									
			r	e	o	1	0	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	5	6	7	8	3	4	5	6	7	8	4	5	6	7	8	≥ 1	13	≥ 14	
j	i	o	1	0	1	2	1	2	1	2	3	4	1	2	3	4	1	2	3	4	5	6	7	8	3	4	5	6	7	8	4	5	6	7	8	≥ 1	13	≥ 14				
inst-1	11	6	5	2	2	4	3	4	3	5	4	5	4	5	2	3	2	3	3	4	2	3	3	4	2	3	2	3	3	3	4	3	3	3	4	4	2	2	2			
inst-2	8	5	5	3	3	4	3	4	3	4	3	5	5	3	3	3	3	3	3	3	3	4	3	4	4	4	3	3	3	3	3	3	3	3	3	3	3	3	3	3		
inst-3	8	5	5	2	2	4	2	4	3	4	3	4	5	2	3	2	2	2	3	2	3	3	4	3	3	4	4	2	3	2	2	3	3	3	3	3	3	3	3	3		
inst-4	10	6	5	2	2	4	2	4	3	4	3	5	5	2	3	2	2	2	3	2	3	3	4	3	3	4	5	2	2	2	3	2	2	3	3	3	3	3	3	3		
inst-5	9	6	5	2	2	4	3	4	3	4	3	5	5	2	3	2	3	3	3	2	3	3	4	3	3	4	4	2	2	2	3	3	2	2	3	3	3	3	3	3		

eleven steps (inst-1), which shows that ignoring presorted input wastes a lot of potential for improvement.

If only small amounts of recovery action ($k = 1$) are allowed, for $j = 1$ the schedule length does not exceed \underline{h} for inst-1 and inst-5 with $p = 0$, for inst-3 and inst-4 with $p \leq 1$, and for inst-2 even for $p \leq 3$, so yet for lowest degrees of recovery we obtain some robustness without increasing the length beyond that of an optimal non-robust schedule. Raising the degree of disturbance to $j \geq 2$, we still obtain a length $h = 4 < \bar{h}$ if the value of p is increased to $p = 1$ for inst-1, to $p = 2$ for inst-2, inst-4, and inst-5, and even to $p = 3$ for inst-3. These values are significantly smaller than those for the strictly robust methods of geometric or even triangular sorting.

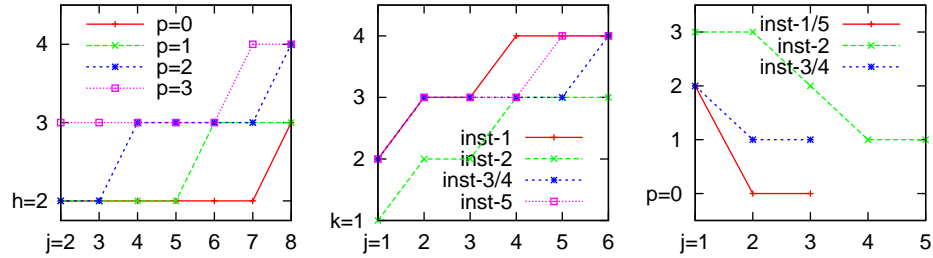


Fig. 2. left: optimal schedule lengths h of inst-1 for $k = 3$; center: highest possible values of k to achieve a length of \underline{h} for $p = 2$; right: smallest possible values of p to achieve a length of \underline{h} for $k = 2$;

The degree of robustness grows rapidly with increasing degrees of recovery, and for $k = 4$ with $p \leq \underline{h}$ —except for inst-4 with $p = 2$ —we can allow any number of delayed trains and still achieve the length \underline{h} of an optimal robust schedule. Between these extremes, Tab. 1 shows how far the value of p can be raised for $k = 2$ and arbitrarily high amounts of delay $j \geq 4$: most instances allow $p = 1$ to obtain $h = 3$, and $h = 4$ can be achieved even for $p = 4$ for three out of five instances. For $k = 3$, Fig. 2 (left) summarizes the values of inst-1: a

schedule of length 3 with a recovery action starting after the third sorting step suffices to cope with a delay of up to six trains and $p = 1$ allows $h \leq 3$ even for any disturbance value j . Similarly, for a fixed value of $p = 2$, Fig. 2 (center) shows the rapid growth of robustness: except for inst-2, k must be raised rather quickly between $j = 1$ and $j = 3$ to achieve a length of \underline{h} , whereas the required value of k does not exceed four for higher disturbances $j \geq 6$. Conversely, Fig. 2 (right) fixes $k = 2$ and shows the maximum value of p that allows a length of \underline{h} : $j = 1$ still allows $p = \underline{h}$ for all instances, but, except for inst-2, this length \underline{h} cannot be achieved for any choice of p for high amounts of delay $j \geq 4$. Hence, higher values of k contribute much more to the potential of recovery than low values of p . Summarizing, through adjusting the recovery parameters k and p , our algorithm presents a tool to flexibly trade off between fast classification and robust schedules and, even for high degrees of robustness, we achieve much shorter schedules than the triangular method currently applied in practice.

5 Conclusion

We have developed a practically applicable algorithm for deriving robust train classification schedules of minimum length. In contrast to [2], we regard multiple inbound and outbound trains, which allows integrating the most relevant disturbance in form of delayed trains. We have introduced the natural recovery action of (p, k) -extensions, for which we proved that the problem is **NP**-complete for every constant $k \geq 1$. Nevertheless, for the simple yet quite general set of scenarios S_j , we have shown our generic algorithm of Sect. 3 can be implemented in polynomial time by solving the subproblem of calculating a maximum recoverable set of breaks efficiently. The experimental study of Sect. 4 indicates that the resulting algorithm improves on the current classification practice as it yields shorter schedules and still allows high degrees of robustness. Its flexibility further allows balancing between strictly robust and optimal non-robust schedules and raises potential for increased traffic throughput in classification yards.

Future Work. Further practical restrictions, such as a limited number of classification tracks (see [16]), are desirable to be considered in the context of robustness. Moreover, the number of cars rolled in presents a secondary objective, which can be additionally minimized for a minimum length. Finally, making the order of inbound trains part of the optimization yields a different robust optimization problem.

References

1. Cacchiani, V., Caprara, A., Galli, L., Kroon, L., Mároti, G.: Recoverable robustness for railway rolling stock planning. In: ATMOS-08. IBFI, Schloss Dagstuhl (2008)
2. Cicerone, S., D’Angelo, G., Di Stefano, G., Frigioni, D., Navarra, A.: Robust algorithms and price of robustness in shunting problems. In: Liebchen, C., Ahuja, R.K., Mesa, J.A. (eds.) ATMOS-07. pp. 175–190. IBFI, Schloss Dagstuhl (2007)

3. Cicerone, S., D'Angelo, G., Di Stefano, G., Frigioni, D., Navarra, A., Schachtebeck, M., Schöbel, A.: Recoverable robustness in shunting and timetabling. In: Ahuja, R., Möhring, R., Zaroliagis, C. (eds.) *Robust and Online Large-Scale Optimization*, LNCS, vol. 5868, pp. 28–60. Springer (2009)
4. Cicerone, S., Di Stefano, G., Schachtebeck, M., Schöbel, A.: Dynamic algorithms for recoverable robustness problems. In: *ATMOS-08*. IBFI, Schloss Dagstuhl (2008)
5. Daganzo, C.F., Dowling, R.G., Hall, R.W.: Railroad classification yard throughput: The case of multistage triangular sorting. *Transp. Res.* 17A(2), 95–106 (1983)
6. Dahlhaus, E., Manne, F., Miller, M., Ryan, J.: Algorithms for combinatorial problems related to train marshalling. In: *AWOCA-00*. pp. 7–16 (2000)
7. Flandorffer, H.: Vereinfachte Güterzugbildung. *ETR RT* 13, 114–118 (1953)
8. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman (1979)
9. Gatto, M., Maue, J., Mihalak, M., Widmayer, P.: Shunting for dummies: An introductory algorithmic survey. In: Ahuja, R., Möhring, R., Zaroliagis, C. (eds.) *Robust and Online Large-Scale Optimization*, LNCS, vol. 5868, pp. 310–337. Springer (2009)
10. Hansmann, R.S., Zimmermann, U.T.: Optimal sorting of rolling stock at hump yards. In: *Mathematics - Key Technology for the Future: Joint Projects Between Universities and Industry*, pp. 189–203. Springer (2007)
11. Hauser, A., Maue, J.: Experimental evaluation of approximation and heuristic algorithms for sorting railway cars. In: Festa, P. (ed.) *SEA-10*. LNCS, vol. 6049, pp. 154–165. Springer (2010)
12. Jacob, R., Márton, P., Maue, J., Nunkesser, M.: Multistage methods for freight train classification. In: Liebchen, C., Ahuja, R.K., Mesa, J.A. (eds.) *ATMOS-07*. pp. 158–174. IBFI, Schloss Dagstuhl (2007)
13. Jacob, R., Márton, P., Maue, J., Nunkesser, M.: Multistage methods for freight train classification. *NETWORKS—Special Issue: Optimization in Scheduled Transportation Networks* (2010), to appear
14. Krell, K.: Grundgedanken des Simultanverfahrens. *ETR RT* 22, 15–23 (1962)
15. Liebchen, C., Lübbecke, M.E., Möhring, R.H., Stiller, S.: The concept of recoverable robustness, linear programming recovery, and railway applications. In: Ahuja, R., Möhring, R., Zaroliagis, C. (eds.) *Robust and Online Large-Scale Optimization*, LNCS, vol. 5868, pp. 1–27. Springer (2009)
16. Márton, P., Maue, J., Nunkesser, M.: An improved classification procedure for the hump yard Lausanne Triage. In: Clausen, J., Di Stefano, G. (eds.) *ATMOS-09*. IBFI, Schloss Dagstuhl (2009)
17. Pentinga, K.: Teaching simultaneous marshalling. *Railway Gaz* pp. 590–593 (1959)
18. Siddiquee, M.W.: Investigation of sorting and train formation schemes for a railroad hump yard. In: *Proc. of the 5th Int. Symposium on the Theory of Traffic Flow and Transportation*. pp. 377–387 (1972)

A Proofs Omitted in Sect. 3

Proof (Lemma 1). (\Rightarrow) Let $X' = \{(\tau_1, \tau_1+1), \dots, (\tau_t, \tau_t+1)\}$ the set of unresolved breaks w.r.t. S within a block of B . Assume $t > 2^k - 1$. For every $(\tau_i, \tau_i+1) \in X'$, schedule B satisfies $b^{\tau_i} = b^{\tau_i+1}$, so, for every (p, k) -extension $\bar{B} = \bar{b}_{h-1+k} \dots \bar{b}_0$ of B , the leading part and the trailing part of \bar{b}^{τ_i} and \bar{b}^{τ_i+1} are equal. For \bar{B} to be a valid schedule w.r.t. scenario S , $\text{num}(\bar{b}_{p-1+k}^{\tau_i} \dots \bar{b}_p^{\tau_i}) < \text{num}(\bar{b}_{p-1+k}^{\tau_i+1} \dots \bar{b}_p^{\tau_i+1})$ for every unresolved break (τ_i, τ_i+1) , $i = 1, \dots, t$. Since also $\text{num}(\bar{b}_{p-1+k}^{\tau_1} \dots \bar{b}_p^{\tau_1}) \geq 0$ and further $\bar{b}_{p-1+k}^{\tau_{i+1}} \dots \bar{b}_p^{\tau_{i+1}} \geq \bar{b}_{p-1+k}^{\tau_i} \dots \bar{b}_p^{\tau_i}$ must hold for every $i = 1, \dots, t-1$, we obtain $\bar{b}_{p-1+k}^{\tau_{t+1}} \dots \bar{b}_p^{\tau_{t+1}} \geq t > 2^k - 1$, which is a contradiction. Hence, there is no (p, k) -extension of B that is valid w.r.t. S if $t > 2^k - 1$.

(\Leftarrow) Let κ denote the number of blocks of B , $\bar{\tau}_i$ the head of the i th block, and λ_i the size of the i th block, $i = 1, \dots, \kappa$. Let further $X'_i = \{(\tau_1^i, \tau_1^i+1), \dots, (\tau_{t_i}^i - 1, \tau_{t_i}^i+1)\} \subseteq X'$ be the subset of unresolved breaks of the i th block with t_i denoting their number.

We extend B in the following way to a schedule B' : for the i th block, we put $b_{p-1+k}^{\tau} \dots b_p^{\tau} = [j]_2$ for every car τ with $\tau_j^i + 1 \leq \tau \leq \tau_{j+1}^i$, where $\tau_0^i + 1 := \bar{\tau}_i$ is the head and $\tau_{t_i+1}^i := \bar{\tau}_i + \lambda_i - 1$ the last car of the i th block. If $|[j]_2| < k$, we simply add leading zeros to obtain a bitstring of length k . Since $j < t_i < 2^k - 1$, also $|[j]_2| \leq k$, so the construction yields a (p, k) -extension of B .

It remains to show that this B' is feasible w.r.t. S . First, if $(\tau_j^i, \tau_j^i+1) \in X'_i$, then $b^{\tau_j^i} = b^{\tau_j^i+1}$ in the original schedule B . The construction yields $b_{p-1+k}^{\tau_j^i} \dots b_p^{\tau_j^i} = [j-1]_2 < [j]_2 = b_{p-1+k}^{\tau_j^i+1} \dots b_p^{\tau_j^i+1}$, so $b^{\tau_j^i} < b^{\tau_j^i+1}$ holds for B' . Second, if $(\tau, \tau+1) \in X \setminus X'$, where X denotes the set of all breaks induced by S , then $b^{\tau} < b^{\tau+1}$ in B . If, in this case, b^{τ} and $b^{\tau+1}$ are in the same block, then $b_{p-1+k}^{\tau} \dots b_p^{\tau} = b_{p-1+k}^{\tau+1} \dots b_p^{\tau+1}$, so $b^{\tau} < b^{\tau+1}$ holds for B' . Otherwise, let b^{τ} be contained in the i th block and $b^{\tau+1}$ in the $(i+1)$ th; then, they satisfy $b_{>p}^{\tau} < b_{>p}^{\tau+1}$, so $b^{\tau} < b^{\tau+1}$ for B' . Finally, if $(\tau, \tau+1) \notin X$, then $b^{\tau} \leq b^{\tau+1}$. If b^{τ} and $b^{\tau+1}$ are in the same block, $b_{p-1+k}^{\tau} \dots b_p^{\tau} = b_{p-1+k}^{\tau+1} \dots b_p^{\tau+1}$, so $b^{\tau} \leq b^{\tau+1}$ still holds for the extension. Otherwise, let b^{τ} be contained in the i th and $b^{\tau+1}$ in the $(i+1)$ th block, $i \in \{1, \dots, i_{\max} - 1\}$; then, $b_{>p}^{\tau} < b_{>p}^{\tau+1}$ already for B , so $b^{\tau} < b^{\tau+1}$ also holds for B' . Therefore, the extension B' presents a feasible schedule w.r.t. S .

Proof (Thrm. 1). Let \bar{B} denote a schedule of κ blocks computed by Alg. 1 and $\bar{\tau}_1, \dots, \bar{\tau}_{\kappa}$ be the heads of the blocks. The unresolved breaks of any block of \bar{B} present an k -recoverable set of breaks by construction and all original breaks are considered. It follows that \bar{B} is a feasible (p, k) -recovery robust algorithm.

Assume, for contradiction, that \bar{B} is not optimal. Let B^* be an optimal (p, k) -recovery robust schedule, κ^* its number of blocks, and $\tau_1^*, \dots, \tau_{\kappa^*}^*$ their heads. Let $\bar{\tau}_i \neq \tau_i^*$, $i \in \{0, \dots, \kappa\}$, with $\bar{\tau}_j = \tau_j^*$ for all $j < i$, and w.l.o.g. let B^* be an optimal schedule that *maximizes* the value of i . We distinguish two cases.

Case 1: $\bar{\tau}_i > \tau_i^*$. Replacing the bitstrings $b^{*\tau_{i-1}^*}, \dots, b^{*\tau_i^*}$ in B^* by the bitstrings $\bar{b}^{\tau_{i-1}^*}, \dots, \bar{b}^{\tau_i^*}$ of \bar{B} produces an optimal schedule, in which the head of the i th block is given by $\bar{\tau}_i$. This contradicts the choice of B^* .

Case 2: $\bar{\tau}_i < \tau_i^*$. Let Y_{i-1}^* be the set of unresolved breaks of the $(i-1)$ th block of B^* , and \bar{Y}_{i-1} the set of unresolved breaks of the $(i-1)$ th block of \bar{B} . Since \bar{Y}_{i-1} is a maximum k -recoverable break set, its size satisfies $|\bar{Y}_{i-1}| \geq |Y_{i-1}^*|$. Regard the number of resolved breaks of B^* in its $(i-1)$ th block:

$$|X_{(\bar{\tau}_{i-1}, \tau_i^*)}| - |Y_{i-1}^*| \geq |X_{(\bar{\tau}_{i-1}, \tau_i^*)}| - |\bar{Y}_{i-1}| > |X_{(\bar{\tau}_{i-1}, \bar{\tau}_i)}| - |\bar{Y}_{i-1}| = 2^p - 1$$

This is a contradiction since there are at most 2^p different bitstrings in a block. As a consequence \bar{B} is optimal.

Proof (Thrm. 2, Part $k = 1$). A given break set X' is 1-recoverable by definition iff there is at most one unresolved break in X' for any scenario. Therefore, this problem is in **NP**.

We reduce from the independent set problem, well known to be **NP**-hard [8]. Let $G = (V, E)$, $V = \{1, \dots, n\}$, be a graph in which we look for an independent set of size a . W.l.o.g., assume G contains no isolated vertex and $|V| - a = 2^p - 1$. In the corresponding 1-recoverable break set instance there are $n+1$ cars. Furthermore, for every edge $(i, j) \in E$ the scenario set contains a scenario S that induces the set of breaks $X^S = \{(i, i+1), (j, j+1)\}$. Note that the size of $X = \cup_{S \in \mathcal{S}} X^S$ and the number of scenarios \mathcal{S} is polynomial in the size of G . We show that there is a recoverable break set $X' \subseteq X$ of size a iff there is an independent set of size a in G .

Let A be an independent set in G with $|A| = a$. Define the break set $X_A = \{(i, i+1) | i \in A\}$. For any scenario $S \in \mathcal{S}$ with $X^S = \{(i, i+1), (j, j+1)\}$, there is an edge $\{i, j\} \in E$, and either $i \notin A$ or $j \notin A$. For this reason at most one of the breaks of X^S is an element of X_A , and thus X_A is a 1-recoverable break set of size a .

Conversely, let X' be a 1-recoverable break set of size a . Define $A = \{i | (i, i+1) \in X'\}$. For any edge $\{i, j\} \in E$ there is a scenario $S \in \mathcal{S}$ with $X^S = \{(i, i+1), (j, j+1)\}$. Since either $(i, i+1) \notin X'$ or $(j, j+1) \notin X'$, at most one endpoint of the edge is in A . Therefore, A is an independent set of size a .

Proof (Thrm. 2, Part $k \geq 2$). We reduce from 2^kSAT , which is strongly **NP**-complete [8]. Let I be an instance of 2^kSAT with variables x_1, \dots, x_n and clauses $\mathcal{C} = \{C_1, \dots, C_m\}$. Each clause contains 2^k different literals. For each x_i we add auxiliary variables $x_{1,i}, \dots, x_{2^{k-1},i}$ and a clause $\bar{C}_i = x_i \vee \bar{x}_i \vee x_{1,i} \vee \bar{x}_{1,i} \vee \dots \vee x_{2^{k-1}-1,i} \vee \bar{x}_{2^{k-1}-1,i}$ to the instance. The new instance is a yes-instance iff the original instance is.

In the k -recoverable break set instance there are $2n + n(2^k - 2) + 1$ cars. For each clause C_j we construct a scenario S_j with $\beta_i \in X^{S_j}$ iff $x_i \in C_j$ and $\bar{\beta}_i \in X^{S_j}$ iff $\bar{x}_i \in C_j$ and define in the same way for each clause \bar{C}_j a scenario \bar{S}_j . Hence, there are $m + n$ scenarios and $2n + n(2^k - 2)$ breaks. Since k is constant, the size of the instance is polynomial in the size of the input data. We will show that there exists a k -recoverable break set $X' \subseteq \cup_{S \in \mathcal{S}} X^S$ with exactly $n + n(2^k - 2)$ breaks iff I is a yes-instance.

Let X' be a k -recoverable break set with $|X'| = n + n(2^k - 2)$. W.l.o.g., all breaks $\beta_{i,j}$ and $\bar{\beta}_{i,j}$ are contained in X' . Otherwise, replace β_i by $\beta_{i,j}$ and $\bar{\beta}_i$ by

$\bar{\beta}_{i,j}$ and get a k -recoverable break set again. Due to $|\cup_{S \in \mathcal{S}} X^S| - |X'| = n$ and the definition of \bar{C}_i , the set X' contains either β_i or $\bar{\beta}_i$ but never both. Thus, the following assignment is well-defined: $x_i^* = \text{true}$ if $\beta_i \in X'$ and $x_i^* = \text{false}$ if $\bar{\beta}_i \in X'$. Since every scenario S_j contains 2^k breaks, at least one break in X^{S_j} is resolved, i.e., there exists an $\ell \in X^{S_j}$ such that β_ℓ or $\bar{\beta}_\ell$ is not an element of X' . If $\beta_\ell \in X^{S_j}$ is resolved, the clause C_j contains the literal x_ℓ and $x_\ell^* = \text{true}$ verifies C_j . If $\bar{\beta}_\ell \in X^{S_j}$ and is resolved, the clause C_j contains the literal \bar{x}_ℓ and $x_\ell^* = \text{false}$. Hence, x^* verifies every clause.

Conversely, let x^* be an assignment which satisfies the 2^k SAT-instance I . If $x_i^* = \text{true}$, we delete β_i from $\cup_{S \in \mathcal{S}} X^S$, and, if $x_i^* = \text{false}$, we delete $\bar{\beta}_i$ to construct a maximum k -recoverable break set X' . The set X' contains $n + n \cdot (2^k - 2)$ elements. Furthermore, in every scenario S_j at most $2^k - 1$ breaks are unresolved. Therefore, X' is a k -recoverable breaks set.

Proof (Cor. 1). A given schedule B is (p, k) -recoverable iff there are at most $2^k - 1$ unresolved breaks in every block of B for any scenario by Lemma 1. Therefore, the problem is in **NP**.

We reduce from the decision version of the maximum k -recoverable break set problem. Let \mathcal{S} be a set of scenarios in which each scenario induces breaks between the cars $1, \dots, n$, $X = \cup_{S \in \mathcal{S}} X^S$, and $K \geq 0$ some constant. W.l.o.g., assume $|X| - K = 2^h - 1$ for some $h \in \mathbb{N}$. Obviously, there exists a feasible (h, k) -recovery robust schedule of length h for this instance iff there is a k -recoverable break set X' of size K in X .

B Proofs Omitted in Sect. 4

Proof (Thrm. 3). Let $S \in \mathcal{S}_j$ and let T_{i_1}, \dots, T_{i_j} be the trains delayed by S . Consider the (unique) scenario $\bar{S} \in \bar{\mathcal{S}}_j$ defined by the trains T_{i_1}, \dots, T_{i_j} and the corresponding permutation $\bar{\sigma}$ of \bar{S} . Now, let $\beta = (\tau, \tau+1) \in X^S$ be a break induced by S . If β is an original break, it will be resolved by B . Otherwise, β is a potential break, so $\tau \in T_x$, $\tau+1 \in T_y$ for some $x \in \{i_1, \dots, i_j\}$ and $x < y \leq \ell$. If $y \in \{i_1, \dots, i_j\}$, then $\bar{\sigma}(x) > \bar{\sigma}(y)$ since $x < y$; if $y \notin \{i_1, \dots, i_j\}$, then $\bar{\sigma}(y) \leq \ell - j < \bar{\sigma}(x)$. In either case $\beta \in X^{\bar{S}}$, so $X^S \subseteq X^{\bar{S}}$. Hence, B is recovery robust for \mathcal{S}_j .

Proof (Lemma 2). Let σ be the permutation defined by S and $\tau+1 \in T_y$, $x < y \leq \ell$. If T_x is delayed and T_y as well, $\sigma(x) > \sigma(y)$ as $x < y$; if T_x is delayed but not T_y , then $\sigma(x) > \ell - j \geq \sigma(y)$. In either case, delaying T_i induces β .

Conversely, if T_x is not delayed but T_y is, $\sigma(x) \leq \ell - j < \sigma(y)$ and β is not induced; if only T_x is delayed, $\sigma(x) < \sigma(y)$ as $x < y$, so β is not induced.

Proof (Thrm. 4). Let $Y_i := X_i \cap X$ be the breaks of X induced by T_i , $i = 1, \dots, \ell$, and let $\bar{X} = \bigcup_{i=1}^{\ell} X'_i$ be the set returned by Alg. 2 for the input Y_1, \dots, Y_ℓ .

To show \bar{X} is a feasible solution, let $S \in \bar{\mathcal{S}}_j$ be a scenario delaying T_{k_1}, \dots, T_{k_j} , i.e., S induces the set of breaks $Y^S := \bigcup_{t=1}^j Y_{k_t} \subseteq X$. Consider the number of

unresolved breaks induced by S : $|\bar{X} \cap Y^S| = \sum_{t=1}^j |\bar{X} \cap Y_{k_t}| = \sum_{t=1}^j |X'_{k_t}| \stackrel{(*)}{\leq} \sum_{t=1}^j |X'_{i_t}| \leq 2^k - 1$, where $(*)$ holds since the ordering in line 1 is invariant for the loop of Alg. 2. Therefore, all unresolved breaks of X induced by S can be recovered, so X' is k -recoverable.

We continue with the optimality and define $X' = \bigcup_{t=1}^j X'_{i_t}$. In case $|X'| < 2^k - 1$, the while-loop (line 2) has never been entered and $X' = X$, so X' is optimal. Otherwise, $|X'| = 2^k - 1$ since, in every iteration of the while-loop, $|X'|$ decreases by exactly one until $|X'| \leq 2^k - 1$. Moreover, as soon as the condition in line 2 is satisfied, $|X'_{i_t}| = |X'_{i_j}|$ for all sets X'_{i_t} with $t \geq j$ that have been decreased at least once in an execution of the loop. For this case, assume there is some k -recoverable set $Z \subseteq X$ with $|Z| > |\bar{X}|$ and define $Z_i := Y_i \cap Z$, $i = 1, \dots, \ell$. Consider the (invariant) order $|X'_{i_1}| \geq \dots \geq |X'_{i_\ell}|$ in line 1 of Alg. 2. Since Z is k -recoverable, $\sum_{t=1}^j |Z_{i_t}| \leq 2^k - 1$.

W.l.o.g. assume $\sum_{t=1}^j |Z_{i_t}| = 2^k - 1$: otherwise, there are some $a \leq j$, $b > j$ with $|X'_{i_a}| > |Z_{i_a}|$, $|X'_{i_b}| < |Z_{i_b}|$; define $\hat{Z}_{i_a} := Z_{i_a} + \beta_1$, $\hat{Z}_{i_b} := Z_{i_b} - \beta_2$ for some $\beta_1 \in X'_{i_a} \setminus Z_{i_a}$, $\beta_2 \in Z_{i_b} \setminus X'_{i_b}$ and $\hat{Z} := Z + \beta_1 - \beta_2$. It remains to show the feasibility of \hat{Z} . Let $S \in \bar{S}_j$ be any scenario delaying T_{i_a} but not T_{i_b} , so $|\hat{Z} \cap X^S| - 1 = |Z \cap X^S|$, and assume, for contradiction, that $|\hat{Z} \cap X^S| = 2^k$. Then, consider Z for the scenario given by S but delaying T_{i_b} instead of T_{i_a} : $|Z \cap X^S| + |Z_{i_b}| - |Z_{i_a}| \geq (|\hat{Z} \cap X^S| - 1) + (|X'_{i_b}| + 1) - (|X'_{i_a}| - 1) = 2^k - 1 + |X'_{i_b}| - |X'_{i_a}| + 2 \geq 2^k$ as $|X'_{i_b}| \geq |X'_{i_a}| - 1$. By this contradiction, \hat{Z} is k -recoverable and $\sum_{t=1}^j |\hat{Z}_{i_t}| > \sum_{t=1}^j |Z_{i_t}|$. Repeating this construction eventually yields a maximum k -recoverable set \hat{Z} with $\sum_{t=1}^j |\hat{Z}_{i_t}| = 2^k - 1$.

Now, let $a \in \{1, \dots, j\}$ be such that $|Z_{i_a}| \leq |Z_{i_t}|$ for all $t \in \{1, \dots, j\}$, and let $b := \{j < t \leq \ell : |Z_{i_t}| > |X'_{i_t}|\}$. Consider the scenario $S \in \bar{S}_j$ delaying T_{i_b} , T_{i_1}, \dots, T_{i_j} except for T_{i_a} . As X'_{i_b} must have been decreased during the execution of Alg. 2, $|Z_{i_b}| > |X'_{i_b}| = |X'_{i_j}| \geq |Z_{i_a}|$ and thus $|X^S| = \sum_{t=1}^j |Z_{i_t}| - |Z_{i_a}| + |Z_{i_b}| > \sum_{t=1}^j |Z_{i_t}| + 1 > 2^k - 1$. By this contradiction Z cannot be k -recoverable and therefore X' is optimal.

The loop in Alg. 2 is executed at most $n - 1$ times since $\sum_{t=1}^j |X_{i_t}| \leq n - 1$, and one execution takes $\mathcal{O}(\log \ell)$. Outside the loop, the sorting in line 1 takes $\mathcal{O}(\ell \log \ell)$, so the total running time is $\mathcal{O}(n \log \ell)$ and thus polynomial.

C Additional Data

Table 2 lists the real-world instances on which we evaluated the algorithm of Sect. 4. The conversion to distinct cars works as in the following example: if $\tau_3 = \tau_6 = 3$, $\tau_7 = \tau_9 = \tau_{10} = 4$, and (τ_{12}, τ_3) as well as (τ_{10}, τ_2) formed original breaks, we put $\tau_3 = 3$, $\tau_6 = 4$, $\tau_7 = 5$, $\tau_9 = 6$, $\tau_{10} = 7$, and further $\tau_2 = 8$.

Table 2. The five problem instances corresponding to five traffic days: n (volume), ℓ (no. of inbound trains), m (no. of outbound trains), β_{\max} : max. no. of breaks of outbound trains, β : total no. of breaks.

instance	n	ℓ	m	β_{\max}	β
inst-1	486	49	24	3	28
inst-2	329	44	24	4	24
inst-3	310	47	24	3	25
inst-4	364	44	24	3	25
inst-5	368	44	27	3	25