Computers and Operations Research 96 (2018) 131–140

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

The vehicle rescheduling problem with retiming

Rolf van Lieshout[a,b,*], Judith Mulder[a], Dennis Huisman[a,b]

[a] Econometric Institute, Erasmus University Rotterdam, 3000 DR Rotterdam, The Netherlands
[b] Erasmus Center for Optimization in Public Transport (ECOPT), Rotterdam, The Netherlands

ARTICLE INFO

Article history:
Received 17 January 2017
Revised 9 April 2018
Accepted 9 April 2018
Available online 10 April 2018

Keywords:
Vehicle rescheduling
Retiming
Lagrangian heuristic

ABSTRACT

When a vehicle breaks down during operation in a public transportation system, the remaining vehicles can be rescheduled to minimize the impact of the breakdown. In this paper, we discuss the vehicle rescheduling problem with retiming (VRSPRT). The idea of retiming is that scheduling flexibility is increased, such that previously inevitable cancellations can be avoided. To incorporate delays, we expand the underlying recovery network with retiming possibilities. This leads to a problem formulation that can be solved using Lagrangian relaxation. As the network gets too large, we propose an iterative neighborhood exploration heuristic to solve the VRSPRT. This heuristic allows retiming for a subset of trips, and adds promising trips to this subset as the algorithm continues. Computational results indicate that the heuristic performs well. While requiring acceptable additional computation time, we find improvements over solutions that do not allow retiming in 58% of the tested instances. By delaying only one or two trips with on average 6–8 min, the average number of cancelled trips is reduced from 1.2 to 0.5.

1. Introduction

Unforeseen events in public transportation systems can lead to severely disrupted schedules, substantially harming passenger satisfaction. Many real time rescheduling methods have been proposed to reduce the impact of disruptions, often tailored to the mode of transport (e.g. bus, air, railway) and the nature of the disruption (e.g. weather conditions, mechanical malfunctions or union strikes). In this paper we consider an extension of the *vehicle rescheduling problem* (VRSP), the problem that originates when a vehicle (bus) in a transportation system breaks down and cannot continue operation. Both the mode of transport of this problem (aircraft and railway rescheduling problems are better researched than bus rescheduling problems) and the nature of its disruption (most papers consider disruptions that only lead to delays) have received considerably less attention in the literature (Bunte and Kliewer, 2009; Visentini et al., 2014).

Li et al. (2004) introduced the VRSP. In Li et al. (2009), the authors reconsider some of their assumptions and propose a new mathematical formulation for the VRSP. More specifically, they no longer exclude the possibility that trips other than the disrupted trip are cancelled and impose penalty costs on rescheduled trips, as it is often desirable that schedule changes are kept to a minimum. Li et al. (2009) propose a Lagrangian heuristic to solve the VRSP. Overall their results indicate that the heuristic performs well, strongly reducing the number of cancellations in comparison with an ad hoc rescheduling procedure.

In this paper, we introduce the vehicle rescheduling problem with retiming (VRSPRT). As retiming, the possibility to delay trips, increases scheduling flexibility, formerly inevitable cancellations can be avoided, an idea that has been successfully demonstrated by Veelenturf et al. (2012) in the crew rescheduling problem. The contribution of this paper is twofold. Firstly, by a clever construction of the underlying recovery network such that it includes all relevant delay options, we can formulate the VRSPRT in a similar way as the VRSP. As a consequence, we can also use a similar Lagrangian heuristic to solve the VRSPRT as the one developed by Li et al. (2009) for the VRSP. The difference between the two heuristics is that the shortest path problems are solved in another network. Secondly, by combining retiming with the idea of neighborhood exploration developed by Potthoff et al. (2010), we are able to find high quality solutions for large problem instances.

The paper is organized as follows. In Section 2, the VRSPRT is thoroughly defined. In Section 3, relevant literature on considering delays in recovery and scheduling problems is discussed. Section 4 presents a mathematical formulation for the VRSPRT. Section 5 presents the Lagrangian heuristic to solve the VRSPRT. Section 6 discusses how the concept of neighborhood exploration can be used to solve large problem instances. In Section 7 the results are presented, illustrating the added value of retiming. We

* Corresponding author.
E-mail addresses: vanlieshout@ese.eur.nl (R. van Lieshout), mulder@ese.eur.nl (J. Mulder), huisman@ese.eur.nl (D. Huisman).

https://doi.org/10.1016/j.cor.2018.04.008
0305-0548/

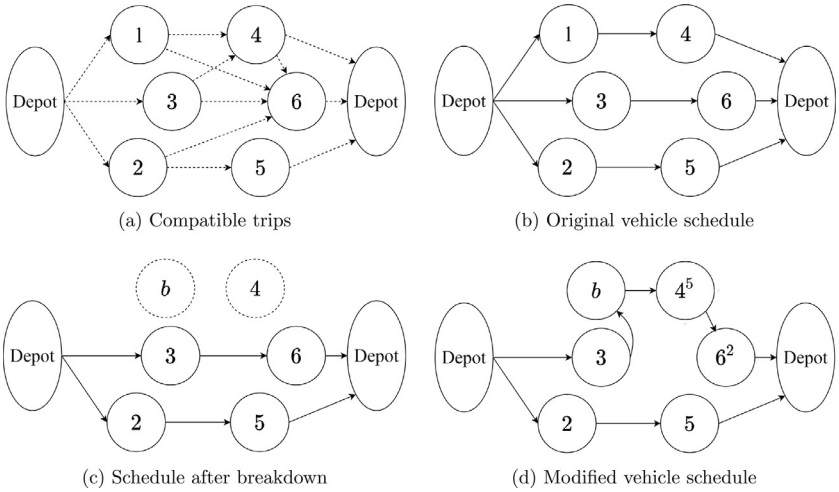

Fig. 1. Example of a scheduling network (a), original schedule (b), schedule after a breakdown of the vehicle performing trip 1 (c) and modified vehicle schedule after rescheduling (d). *b* represents the back-up trip. Superscripts indicate the delays in minutes for the modified schedule.

Table 1
Example of a part of a timetable.

Trip	Starting time	Ending time	Starting location	Ending location
1	9:00	10:15	C	B
2	9:15	10:30	B	C
3	9:27	10:10	A	B
4	10:27	11:10	B	A
5	10:45	11:30	C	A
6	11:13	12:18	A	C

wrap-up the paper with a conclusion and suggestions for further research.

2. Problem description

The VRSP(RT) is defined for a bus transit system where a number of *trips* specified in a predefined timetable need to be performed. In Table 1 an example of a small timetable is shown. A trip consists of multiple stops where passengers are picked up and dropped off and has a specified starting and ending location and a starting and ending time. Every vehicle starts at a depot and executes a sequence of trips before returning to the depot it started at. A pair of trips in a sequence needs to be *compatible*, meaning that a vehicle can perform the second trip after the first trip. Not all pairs of trips are compatible, as the times that they are carried out might overlap or it takes too long to get from the ending location of the first trip to the starting location of the second trip. Movements of a vehicle without passengers, from or to a depot or from the ending location of one trip to the starting location of the next trip, are referred to as *deadheads*.

In Fig. 1a, the scheduling network representing all pairs of compatible trips in Table 1 is visualized. Here, we assume that the deadhead driving time between all locations is 40 min. Fig. 1b depicts an example of a vehicle schedule corresponding to the timetable.

The rescheduling needs to take place when one of the vehicles breaks down at a certain *breakdown time* and *breakdown location*. The trip currently being performed by the disrupted vehicle is referred to as the *cut trip* and the remainder of the trips scheduled for the disrupted vehicle is referred to as the *cut path*. In case the cut trip is a trip, a *back-up vehicle* needs to be sent to the breakdown location to pick up the stranded passengers and finish the remainder of the trip. This task is referred to as the *back-up trip*. In the other case, the cut trip is a deadhead, a back-up vehicle is not necessary. As the latter case gives an easier and less interesting problem, we only consider the case where the cut trip is an actual trip. Fig. 1c depicts the situation where the vehicle originally performing trips 1 and 4 breaks down during trip 1. Node *b* represents the back-up trip.

After the breakdown of one vehicle, the remaining vehicles, which are either at a depot, performing a deadhead or operating a trip, can be seen as *pseudo-depots* from which one vehicle can be deployed at a certain *availability time* from a certain *availability location*. The availability time of vehicles at a depot or performing a deadhead is equal to the breakdown time and the availability location is their current location. On the other hand, as ongoing trips must be finished before rescheduling, the availability time of vehicles performing a trip is equal to the ending time of their current trip, and the availability location is the ending location of their current trip. Moreover, if there are back-up vehicles available at any of the real depots, these can instantly be deployed, such that the availability time of the real depots are also equal to the breakdown time. However, since the problem often becomes trivial when back-up vehicles are available at the depots, we do not consider this option in this paper.

Fig. 1d visualizes an example of a modified vehicle schedule. Here, the vehicle originally performing trip 3 and 6 serves as the back-up vehicle, i.e. after the vehicle has finished trip 3, it drives to the breakdown location, picks up the passengers and finishes trip 1. In the example we assume the back-up vehicle is able to finish the back-up trip at 10:32, such that the vehicle can then perform trip 4 with a delay of 5 min and trip 6 with a delay of 2 min.

In the VRSP, the only allowed rescheduling operation is to reassign trips to different vehicles. In the VRSPRT, it is also allowed to delay trips. Delaying trips is undesirable from a passengers' point of view. Conversely, reassigning trips is undesirable from the point of view of the operator, as the crew of a vehicle might not be familiar with all trips and reassignments might lead to crews having to work overtime. As such, both introduced delays and reassignments should be penalized when rescheduling.

The VRSPRT can now be defined as follows. Given a current schedule, a breakdown time and a breakdown location, find a new feasible schedule that minimizes the sum of cancellation, reassignment and delay costs. Li et al. (2009) also include operation costs in this definition, but as the costs of operating the transit system can be regarded as sunk costs, we decided to focus only on cancellation and reassignments costs.

It must be noted that in contrast with the vehicle scheduling problem (VSP), the computational complexity of the VRSP(RT) does not depend on whether there is a single or multiple depots. In particular, the single-depot vehicle rescheduling problem already is NP-hard, as shown by Li et al. (2009). In this paper, we limit ourselves to the single-depot case to keep the exposition as simple as possible.

3. Literature review

In the context of road-based vehicle rescheduling, Huisman et al. (2004) are the first to take delays into account. They consider a dynamic environment where they reschedule when one or more vehicles have incurred delays. By imposing delay costs on arcs in a multi-commodity flow formulation, delays for future trips can be reduced or avoided. A disadvantage of modeling delays in this manner, is that not all delays on arcs can be computed beforehand, because delays might be propagated to later trips. For instance, if a trip can be executed by two vehicles, one of which is delayed, it is unknown whether delay costs should be imposed. Li et al. (2008) circumvent this problem by modeling delays explicitly in a multi-commodity network flow problem with time windows for the truck schedule recovery problem. However, this requires a large number of additional variables and constraints (one variable and one additional constraint for each trip and one variable and four constraints for each compatible pair of trips), making it impractical for solving medium or large sized instances without a specialized algorithm. The authors solve instances of only 23 vehicles and 31 trips using CPLEX. Trip cancellations are not considered in their research, but could easily be included in the problem formulation. More recently, Dávid and Krész (2017) allow retiming of trips in the dynamic vehicle scheduling problem. In this problem, the vehicle schedule needs to be adjusted in an online manner when disruptions occur. The main difference with the VRSP(RT) is that the authors only consider disruptions where vehicles are only temporarily out-of-service, whereas in the VRSP(RT) the broken down vehicle cannot be operated for the rest of the day (which gives a more difficult problem). Dávid and Krész (2017) present two heuristics that provide high quality solutions on instances with 800 trips. On larger instances the authors are not able to measure the quality of the found solutions.

In the airline industry, flight delays are very costly, explaining why more papers take delays into account in the air-based rescheduling literature. Thengvall et al. (2001) incorporate delays in a multi-commodity flow formulation by adding a series of flight options for each flight that needs to be covered. The delay times are predetermined, so e.g. one flight option represents a 20-min delay and a second option a 60-min delay. All arc costs can be computed beforehand because the formulation is based on a time-space network rather than the connection-based network used by Huisman et al. (2004). On the other hand, the method of Thengvall et al. (2001) potentially leads to redundant arcs and inefficiencies, since perhaps the first option is impossible to be reached or a plane that could depart with a 21-min delay must wait for another 39 min. Other papers employ a set partitioning formulation, where every set represents a certain sequence of flights (trips). The advantage of the set partitioning formulation is that delays can be incorporated without difficulties, as the delay cost of a sequence of flights can be evaluated after the whole sequence is constructed. Løve et al. (2002) develop two neighborhood search heuristics to solve the resulting problem. Stojković and Soumis (2001) and Eggenberg et al. (2010) take a different approach and solve the set partitioning problem using column generation. A difference between the two is that the former represents the recovery network as a connection based network, while the latter uses a time-space network. An advantage of the time-space network is that the time windows of every flight are satisfied in every path in the network. Consequently, the time windows need not be considered in the pricing problem, which is therefore easier to solve. On the other hand, Eggenberg et al. (2010) do not embed the algorithm in a branch-and-bound scheme, so they cannot guarantee optimality.

Railway rescheduling problems are less similar to the VRSP as aircraft rescheduling problems, because the disruption of a train immediately influences surrounding trains since tracks and stations are shared across trains. However, some of the proposed solution approaches are still useful. Potthoff et al. (2010) solve the railway crew rescheduling problem using an iterative approach. As the number of possible duties is extremely large, initially only a subset of all duties is considered. The corresponding problem is referred to as the core problem and solved using column generation. In case some tasks are cancelled in the solution of the core problem, duties that lie in a neighborhood of the uncovered tasks are added to the core problem, after which the procedure is repeated. Veelenturf et al. (2012) extend the railway crew rescheduling problem by introducing retiming. Delay possibilities are incorporated into the problem by introducing copies of tasks that need to be executed, each with a different starting time. The problem is solved by column generation in combination with Lagrangian heuristics.

Retiming has not only been considered in rescheduling problems, but also in scheduling problems. In the (multiple-depot) vehicle scheduling problem with time windows (MDVSPTW), the starting time of every trip is not fixed, but restricted within a predefined time interval. Desaulniers et al. (1998) formulate the MDVSPTW as a multi-commodity flow model with continuous time windows. They solve the model using branch-and-price. Due to the time windows, the column generation subproblem is a shortest path problem with resource constraints. Kliewer et al. (2012) use discrete instead of continuous time windows, meaning that the starting time of a trip can only be shifted by full minutes. The underlying network, represented using a time-space network, is extended with time window arcs representing the shifted options. Furthermore, they remove time window arcs that do not create new connections. Using this technique, all time windows are satisfied in all paths in the underlying network, such that it is not needed to include specific time window constraints in the mathematical formulation. Kliewer et al. (2012) use this modeling technique to solve the MDVSPTW and the crew scheduling problem simultaneously. Their solution approach consists of a combination of column generation and Lagrangian relaxation and is based on Huisman et al. (2005a).

As is clear, many different methods for including delays exist in the literature. However, to the best of our knowledge, no method exists that efficiently models all delay options and can give high quality solutions in a short amount of time, which is necessary for the VRSPRT. Consequently, in this paper we will now propose a new method for incorporating delays in rescheduling problems exhibiting the desired features.

4. Mathematical formulation

Let $N = \{b, n_1, n_2, \ldots, n_m\}$ denote the set of future trips (viewed from the breakdown time). The back-up trip is denoted by b and has a starting time equal to the breakdown time, as it can be performed from the moment of breakdown, and a duration equal to the remaining duration of the cut trip plus a certain service time. Define D as the set of all pseudo-depots, which represent all vehicles at the depot or operating a (deadhead) trip, apart from the disrupted vehicle. Furthermore, s_d is used to denote the source node representing pseudo-depot d and t is used to denote the real depot as an endpoint.

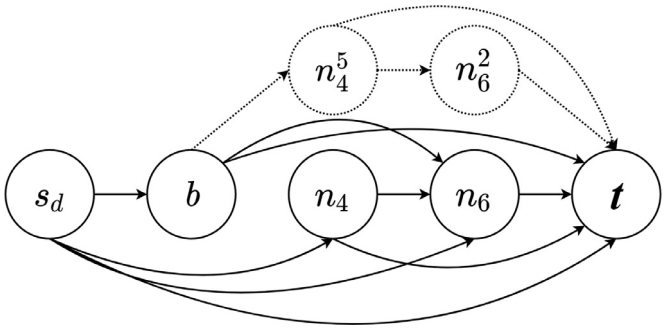

Fig. 2. A recovery network extended with delay options. The dotted part of the network represents the options that become possible by introducing delays.

We use connection networks to represent all rescheduling possibilities from every pseudo-depot. Define N^d as the set of trips that can be performed from pseudo-depot d, and $E^d = \{(i, j) | i < j, i$ and j compatible, $i \in N^d, j \in N^d\}$ as the set of deadheads that can be performed from pseudo-depot d. The recovery network from pseudo-depot d without retiming is then given by $G^d = (V^d, A^d)$, where $V^d = N^d \cup \{s_d, t\}$ and $A^d = E^d \cup (s_d \times N^d) \cup (N^d \times t) \cup (s_d, t)$.

To expand G^d with retiming possibilities, we maintain a list W of all nodes that remain to be explored. To make sure we explore each node only once, we sort the nodes in W according to a topological ordering, which can be achieved by sorting according to scheduled starting times. Initially, all nodes in G^d remain to be explored such that $W = \{s_d, b, n_1, n_2, \ldots, n_m, t\}$. We let $q_{\max}$ denote the maximum allowed delay. Furthermore, we only allow delaying trips with full minutes to limit the size of the network. Note that this discretization is not unreasonable as timetables are usually specified in full minutes.

We start the expansion by exploring s_d, the first node in W, and iterate over all trips j that cannot be operated from s_d without retiming, i.e. $(s_d, j) \notin A^d$. In case j can be operated from s_d if j is retimed with at least q minutes and $q \leq q_{\max}$, we add a copy of j, denoted by n_j^q, to the network. Next to that, we add the arc (s_d, n_j^q) and add arcs from n_j^q to all compatible trips from this node. When all trips j are considered, we remove s_d from W, add the newly created nodes to W, resort the list and proceed by exploring the new first node in W, say node i. Again we iterate over all trips j that cannot be operated from i without retiming. In case j can be operated from i if j is retimed with at least q minutes and $q \leq q_{\max}$, we now first check whether the node n_j^q is already present in the network. If yes, we add an arc to this node. Otherwise, we add n_j^q and add all relevant arcs. When all trips j are considered, i is removed from W, newly created nodes are added and W is resorted. This process continues until the list W is empty.

By only adding arcs from nodes to the earliest compatible retimed copy, we make sure that the network is only expanded if it introduces a new connection. We denote the resulting expanded recovery network of pseudo-depot d by $G_e^d = (V_e^d, A_e^d)$ and the set of all nodes corresponding to trips by N_e^d. We denote the set of nodes corresponding to trip k by $N^d(k)$ and define $A_e^d(k) = \{(i, j) \mid i \in N^d(k)$ and $(i, j) \in A_e^d\}$, the set of arcs leaving nodes associated to trip k. Finally, note that the networks can be efficiently generated and stored by recognizing that the networks for all different pseudo-depots look very much alike. Therefore, instead of maintaining $|D|$ separate networks it is beneficial to maintain a single network and keep track of which arcs can be reached from each pseudo-depot.

An example of an extended recovery network is presented in Fig. 2. This example corresponds to the extended recovery network of the vehicle originally performing trip 3 and 6 in Fig. 1, in the case that the vehicle originally performing trip 1 and 4 breaks down near the end of trip 1. As such, s_d is a pseudo-depot with location B and availability time 10:10 (the ending location and ending time of trip 3). The solid part is the network without retiming and the dotted part is added when delays are introduced. It can be seen that originally, trip 4 could not be performed after the back-up trip. However, if trip 4 is delayed with 5 min (see node n_4^5) it can be performed after the back-up trip, with the side-effect that trip 6 needs to be delayed with 2 min if one wants to execute it after trip 4. Note that even though it is possible that a vehicle performs for example n_6^2 after n_4, this arc is not added to the network as it is simply not necessary to delay trip 6 if trip 4 is performed without delay.

Let c_{ij}^d denote the cost of arc $(i, j) \in A_e^d$, C_k the cost of cancelling trip k. If we introduce decision variables y_{ij}^d, indicating whether arc $(i, j) \in A_e^d$ is selected, and z_k, indicating whether trip k is cancelled, the VRSPRT can be formulated as follows:

$$\min \quad \sum_{d \in D} \sum_{(i,j) \in A_e^d} c_{ij}^d y_{ij}^d + \sum_{k \in N} C_k z_k \tag{1}$$

$$\text{s.t.} \quad \sum_{\{j:(s_d,j) \in A_e^d\}} y_{s_d j}^d = 1 \qquad \forall d \in D, \tag{2}$$

$$\sum_{d \in D} \sum_{(i,j) \in A_e^d(k)} y_{ij}^d + z_k = 1 \qquad \forall k \in N, \tag{3}$$

$$\sum_{\{j:(i,j) \in A_e^d\}} y_{ij}^d - \sum_{\{j:(j,i) \in A_e^d\}} y_{ji}^d = 0 \qquad \forall d \in D, \forall i \in N_e^d, \tag{4}$$

$$y_{ij}^d, z_k \in \{0, 1\} \qquad \forall d \in D, \forall (i, j) \in A_e^d, \forall k \in N. \tag{5}$$

The objective (1) is to minimize the sum of reassignment costs and delay costs (which are both included in the arc costs c_{ij}^d) and cancellation costs. Constraints (2) assure that exactly 1 vehicle departs from every pseudo-depot. Constraints (3) guarantee that every trip is either executed or cancelled. In case the triangle equality holds for the arc costs or the penalty for each reassignment is sufficiently large, the equality sign in these constraints can be replaced by a "$\geq$" sign (Li et al., 2009). As both conditions are likely to hold, we assume this can be done in the remainder of the paper. Constraints (4) are flow conservation constraints.

Note that by expanding the recovery network with retiming options instead of modeling explicitly, we can give a formulation for the VRSPRT that has the same structure as the VRSP. In the next section, we will exploit this similarity to develop a heuristic for the VRSPRT.

5. Lagrangian heuristic

Because of the similar structures of the VRSP and the VRSPRT we can modify the Lagrangian heuristic that Li et al. (2009) developed for the VRSP to solve the VRSPRT. As this heuristic provides high quality solutions in a short amount of time for the VRSP, we expect that it also performs well on the VRSPRT. Moreover, as we will see in Section 6, an additional advantage of using Lagrangian relaxation is that the Lagrangian solutions can be used to identify for which trips retiming is beneficial. In this section we give a description of the heuristic and discuss the adaptations that are needed to apply it to the VRSPRT.

An overview of the Lagrangian heuristic is as follows:

Step 1. Compute a lower bound by solving the Lagrangian problem.

Step 2. Compute an upper bound by applying an insertion-based primal heuristic that transforms the Lagrangian solution into a feasible solution.
Step 3. Update the Lagrangian multipliers based on subgradient search.
Step 4. Repeat steps 1–3 until the gap between the lower and upper bound is sufficiently small or a given time limit is exceeded.

5.1. Solving the Lagrangian

We relax constraints (3), $\sum_{d\in D}\sum_{(i,j)\in A_e^d(k)} y_{ij}^d + z_k \geq 1 \quad \forall k \in N$, and incorporate them in the objective function. The resulting Lagrangian problem can be written as

$$\theta(\boldsymbol{\lambda}) = \sum_{k\in N}\lambda_k + \sum_{k\in N}\kappa_k(\boldsymbol{\lambda}) + \sum_{d\in D}\mu_d(\boldsymbol{\lambda}), \tag{6}$$

where $\kappa_k(\boldsymbol{\lambda})$ and $\mu_d(\boldsymbol{\lambda})$ are sub-problems for trip k and pseudo-depot d respectively. The first sub-problem is given by

$$\kappa_k(\boldsymbol{\lambda}) = \min\,(C_k - \lambda_k)z_k, \;\text{ s.t. } z_k \in \{0,1\} \tag{7}$$

which has the trivial solution that $z_k = 1$ if $C_k - \lambda_k < 0$ and 0 otherwise. For the other sub-problems, let $T(i)$ return the trip $k \in N$ associated with node $i \in N_e^d$ and define

$$o_{ij}^d(\boldsymbol{\lambda}) = \begin{cases} c_{ij}^d, & \text{if } i = s_d \\ c_{ij}^d - \lambda_{T(i)}, & \text{otherwise.} \end{cases}$$

Then, we can write the second sub-problem as

$$\mu_d(\boldsymbol{\lambda}) = \min \sum_{(i,j)\in A_e^d} o_{ij}^d(\boldsymbol{\lambda}) y_{ij}^d \tag{8}$$

$$\text{s.t.} \sum_{\{j:(s_d,j)\in A_e^d\}} y_{s_d j}^d = 1, \tag{9}$$

$$\sum_{\{j:(i,j)\in A_e^d\}} y_{ij}^d - \sum_{\{j:(j,i)\in A_e^d\}} y_{ji}^d = 0 \qquad \forall i \in N_e^d, \tag{10}$$

$$y_{ij}^d \in \{0,1\} \quad \forall (i,j) \in A_e^d, \tag{11}$$

which can observed to be a shortest path problem with transformed arc costs $o_{ij}^d(\boldsymbol{\lambda})$. This problem can be solved very efficiently, especially if we exploit the topological ordering of the graph (Cherkassky et al., 1996).

As can be seen, for a given set of Lagrangian multipliers, $\theta(\boldsymbol{\lambda})$ can be solved by decomposing the problem into 'easy' problems that can be solved by inspection or for which specialized algorithms can be employed.

The Lagrangian multipliers are updated using the update formula from Held and Karp (1971), which is based on subgradient optimization. Based on initial experiments, we use an initial step-size of 7 and half the step-size every 6 iterations.

5.2. Primal heuristic

To generate upper bounds, Li et al. (2009) propose a primal heuristic that computes a feasible solution from the Lagrangian solution. Because constraints (3) are relaxed, a trip might be covered multiple times or might not be covered nor cancelled in the Lagrangian solution. The idea of the primal heuristic is to first remove redundant covering and then insert uncovered trips into existing paths (routes) if this reduces costs. Only if an uncovered trip cannot be inserted in any of the paths or inserting it causes the costs to increase, it is cancelled. A more detailed description of the heuristic is given below.

- **Step 1.** Remove redundant covering for each trip. If a trip is covered more than once, remove the trip from paths other than the cheapest path covering the trip.
- **Step 2.** For each uncovered trip, determine in which paths it can be inserted such that the costs decrease and request an insertion in the path that yields the maximum cost decrease. If no feasible insertion position in any of the paths exists or none of the insertions reduce the costs, cancel it.
- **Step 3.** For every requested insertion position, insert the trip that yields the maximum cost decrease.
- **Step 4.** Repeat steps 2 and 3 until every trip is either covered or cancelled.

The cost decrease of an insertion is given by the difference between the objective function of the VRSPRT before and after the insertion. In this step, we include a very small portion of the operation costs in the objective to break symmetries. During a removal, we need to check whether trips after the removed trip can be performed with less delay. Consider the recovery network from Fig. 2 and assume the back-up trip needs to be removed from the path $\{s_d, b, n_4^5, n_6^2, t\}$. As it is no longer necessary to delay trips 4 and 6, the result is the path $\{s_d, n_4, n_6, t\}$. During an insertion, we need to check whether the inserted trip and trips after the inserted trip need to be delayed to maintain a feasible path. For example, assume that trip 4 needs to be inserted in the path $\{s_d, b, n_6, t\}$. This insertion is feasible only if trip 4 and trip 6 are delayed, hence the result is $\{s_d, b, n_4^5, n_6^2, t\}$.

6. Iterative neighborhood exploration

Even with starting times rounded to full minutes and small allowed delays, the introduction of retiming can lead to a significant increase in the size of the recovery network. For instances of 700 trips, when we permit a maximum delay of only 5 min, the number of arcs and nodes can be more than three times as large compared to the network without retiming. As a result, the Lagrangian heuristic has a very hard time finding good solutions. In five test instances, the heuristic could on average perform only nine iterations in 5 min. Due to this very limited number of iterations, instead of resulting in fewer cancellations, the introduction of retiming lead to solutions with more cancelled trips. As is clear, the size of the network needs to be reduced in order to realize a feasible solution method.

To this end, we apply the ideas of Potthoff et al. (2010) to the VRSPRT. In this paper, the authors consider a subset of all possible duties in the railway crew rescheduling problem. If not all tasks are covered in the resulting solution, duties in the neighborhood of uncovered tasks are added to the duties under consideration, after which the process is repeated. The neighborhood is defined such that duties in the neighborhood can possibly cover the uncovered task. Likewise, in the VRSPRT we can start by only allowing a subset of all trips to be delayed. If the Lagrangian heuristic gives a solution with cancelled trips, certain trips can be added to the subset. An overview of our approach, which we refer to as *iterative neighborhood exploration* (INE), is outlined in Fig. 3.

More specifically, we maintain a set R of trips that are allowed to be retimed. Initially, only the back-up trip is in this set. In other words, we start by solving the VRSP. If the solution of the Lagrangian heuristic does not include any cancellations, the algorithm is terminated. Otherwise, trips in the neighborhood of difficult trips are added to R. To speed up the process, we do not only regard cancelled trips in the final solution of the heuristic as *difficult* trips, but all trips i for which it holds that (i) $z_i = 1$ in the Lagrangian solution (implying that this trip has a high Lagrangian multiplier, so it is difficult to perform) or (ii) is not performed in the Lagrangian solution (before the insertion heuris-

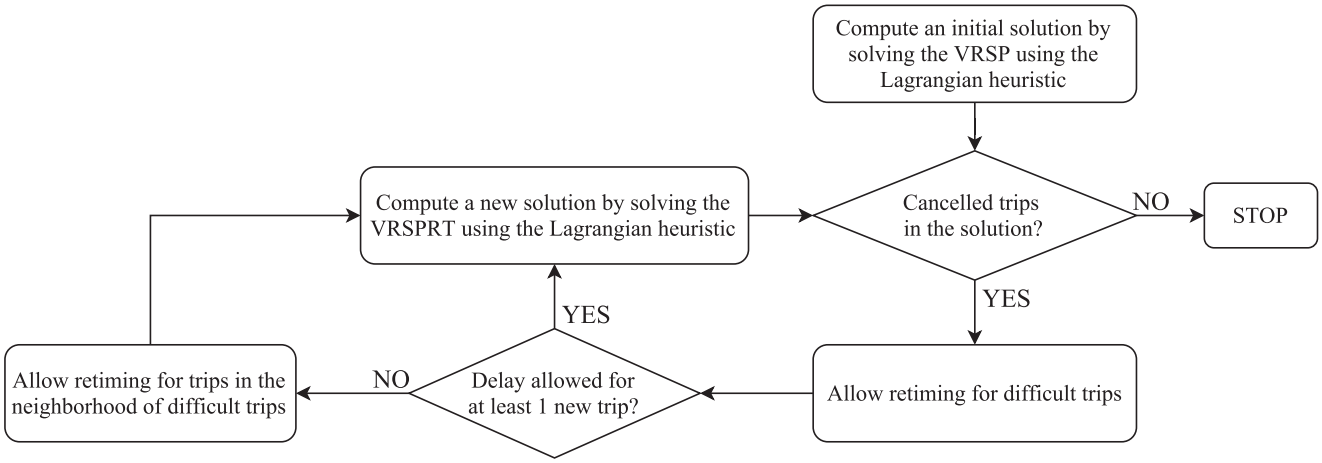

Fig. 3. The iterative neighborhood exploration heuristic for the VRSPRT.

tic). This broader definition enables us to extract more information from the Lagrangian optimization.

As for the neighborhood of a trip, let s_i and e_i denote the scheduled starting and ending time of trip i. Let τ_{ij} denote the deadhead time from the ending location of trip i to the starting location of trip j and let υ_{ij} denote the deadhead time from the starting location of trip i to the starting location of trip j. We define that trip j lies in the neighborhood of trip i if it satisfies one of the following conditions:

1. $|s_j - s_i| < m - \upsilon_{ij}$,
2. $|s_j - e_i| < m - \tau_{ij}$,

where m is a given positive parameter. The intuition for this definition is as follows. If a trip from Rotterdam to Amsterdam starting at 1 p.m. and ending at 3 p.m. is cancelled, this indicates that there is a shortage of vehicles in Rotterdam around 1 p.m. If we allow delays for trips starting near Rotterdam around 1 p.m. (the first condition), perhaps the cancellation can be avoided. Moreover, to prevent missed connections in Amsterdam if the formerly cancelled trip is delayed, delays can also be allowed for trips starting near Amsterdam around 3 p.m. (the second condition).

In our experiments, we initialize m with the value 20 min, as this leads to neighborhoods of appropriate sizes (on average 6 trips). In case all trips in the neighborhoods of difficult trips are already in R, we keep increasing m by 10 min until we find trips in the neighborhood that are not in R. To speed up the algorithm, the Lagrangian multipliers of the iteration of the Lagrangian heuristic that led to the best feasible solution are stored and used as initial values for the Lagrangian multipliers in the next iteration of the algorithm.

7. Computational results

We have performed a computational study to analyse the performance of the INE heuristic and to investigate whether the introduction of retiming can lead to better solutions, especially regarding the number of cancellations. First, we describe the problem instances that we considered. Second, we compare the solutions of the INE heuristic to the solutions of the VRSP. Finally, we test the sensitivity of the solutions with respect to the different cost parameters.

7.1. Experimental setup

Problem instances are generated as in Li et al. (2009). First, a vehicle scheduling instance consisting of 700 trips is generated using the method of Carpaneto et al. (1989). This method aims to simulate a real-life public transport system in which short and long trips are executed from 5 a.m. till midnight. Short trips have a trip duration around 50 min and are more likely to start in peak hours (7–8 a.m. and 5–6 p.m.). Long trips have a duration between 3 and 5 h and have starting times that are distributed uniformly over the whole day. We consider two classes of instances. In class S all trips are short trips and in class M the ratio between short and long trips is 40:60. The vehicle scheduling instance is solved, after which an early short trip is selected as the cut trip. The breakdown occurs 80% into the cut trip. When a back-up vehicle arrives at the breakdown location, it serves the remainder of the cut trip after a service time of 3 min. Moreover, only vehicles that can arrive at the breakdown location within 25 min after breakdown are allowed to serve as back-up vehicles. The recovery network is constructed accordingly.

Table 2
Average characteristics of the used problem instances.

Class	Vehicles	Remaining trips	Trips in cut path
S	109.2	562.3	6.2
M	168.2	567.1	5.0

The penalty costs for cancellations and reassignments are defined in the same way as in Li et al. (2009). That is, a penalty P is added to arcs $(i, j) \in A_e^d$ if the trip corresponding to node j is not originally performed from depot d. The cancellation costs are defined as $C_i = 5t_i + C$, where t_i is the duration of trip i and C is a fixed component. We impose a very large cancellation cost on the back-up trip to make sure it is performed. The delay costs are v per minute and also incorporated in the arc costs. No delay costs are imposed for the back-up trip. Lastly, starting times of trips are rounded to full minutes to limit the number of retiming options.

We generated 50 instances for each class. Average characteristics of the problem instances are presented in Table 2. The average number of vehicles in class M is larger than in class S, because of the longer trips that exist in class M. This also explains why the average number of trips in the cut path is larger for class S, as in class S more trips can be scheduled per vehicle. Note that even though instances in class M might seem easier to solve (there are more vehicles that can be used for rescheduling and fewer trips in the cut path), this turns out not to be the case as many vehicles are performing long trips at the moment of breakdown and can only be rescheduled when this trip is finished.

The maximum delay we allow is 15 min. We perform a maximum of four iterations of the INE heuristic. The time limit of ev-

Table 3
Results of the INE heuristic compared to the heuristic without retiming.

	Without retiming			With retiming				
Class	Cancelled trips	Reassigned trips	Costs	Cancelled trips	Reassigned trips	Delayed trips	Costs	Fewer cancellations due retiming (%)
S	0.92	5.00	4599	0.46	5.42	0.62	3826	40
M	1.44	3.86	5861	0.80	4.36	0.96	4596	54

Table 4
Characteristics of the solutions where the INE heuristic was successful.

Class	Cancelled trips	Reassigned trips	Delayed trips	Average delay (min.)	Cost reduction (%)	$\|R\|$
S	0.18	6.00	1.41	7.58	36	91
M	0.66	5.10	1.66	8.04	36	85

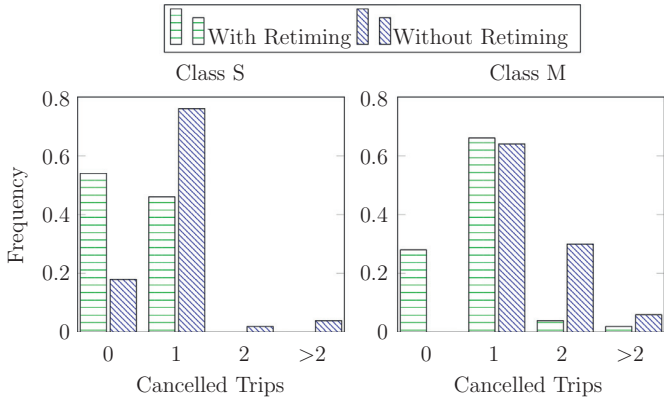

Fig. 4. Histograms of the number of cancellations with and without retiming for class S and class M.

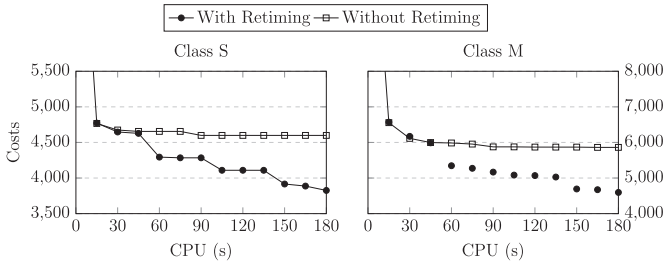

Fig. 5. Plot of the average objective value (of the best found feasible solutions) versus the computation time.

ery iteration is 45 s, such that the total computation time is at most 3 min. To find benchmarks for the INE heuristic, we run the Lagrangian heuristic without retiming for 3 min. The algorithm is implemented in Java on a HP EliteBook 8460p running Windows 10 with an Intel Core i5 processor at 2.5 GHz and 4GB of RAM.

7.2. Performance of the iterative neighborhood exploration heuristic

In this first set of experiments, P is set equal to 500, C to 2000 and v to 10. These values for P and C are also used by Li et al. (2009). In Table 3 we present the average number of cancelled trips, reassigned trips, delayed trip and the average costs of the solutions found by the INE heuristic and by the heuristic without retiming. The INE heuristic finds solutions with fewer cancellations in 40% of the cases in class S and in 54% of the cases in class M. On average, retiming reduces the number of cancelled trips from 0.92 to 0.46 and from 1.44 to 0.80, respectively. To achieve this reduction, on average 0.62 trips in class S and 0.96 trips in class M are delayed, and slightly more trips need to be reassigned. The decrease in objective value when retiming is introduced is partially compensated by extra incurred costs for delays and reassignments, but the decrease is still substantial, at 773 (17%) for class S and 1265 (22%) for class M. In Fig. 4, the histograms for the number of cancelled trips with and without retiming are displayed. It can be seen that the entire distribution of the number of cancelled trips shifts to the left when retiming is introduced. In the solutions of the INE heuristic, very few instances have more than 1 cancelled trip.

From Table 3 and Fig. 4 it can be concluded that the INE heuristic certainly achieves the desired results with respect to the number of cancelled trips. We will now analyse whether the introduced delays to achieve this reduction are not too long. In Table 4 we present the average number of cancelled, reassigned and delayed trips, the average delay in minutes, the average cost reduction and the average number of trips for which retiming was allowed ($|R|$) for the solutions where cancellations were avoided using retiming. As can be seen, the number of delayed trips is moderate; on average only 1.41 trips in class S and 1.66 trips in class M. Furthermore, the length of the delays that are needed to avoid cancellations are certainly reasonable; on average about 8 min. Finally, note that $|R|$ is around 90 in both classes. This shows that using a proper definition of difficult trips and the neighborhood of a trip, we are able to identify the trips for which retiming is really beneficial.

To further illustrate the difference between the INE heuristic and the heuristic without retiming, Fig. 5 depicts the average objective value over all instances against the computation time, with and without retiming. There is no difference between solution values before 45 s as the INE heuristic is initialized by solving the VRSP without retiming. Furthermore, we can observe that the best solutions without retiming are mostly already found within these 45 s. After 45 s, the neighborhood exploration heuristic enters the second iteration, at which point the average costs strongly decrease by introducing retiming. In class S, there are clear jumps every time the heuristic starts a new iteration (in which more trips are allowed to be retimed). In class M, the decrease is more gradual.

As a final illustration of the INE heuristic, we will sketch the solutions obtained after every iteration of one of the instances in class M. The original schedule of the vehicles that turn out to be relevant in the rescheduling process is visualized in Fig. 6a. Nodes with a letter V represent the vehicles and nodes with a number correspond to trips. VB denotes the vehicle that breaks down, and B denotes the trip where the breakdown takes place. Nodes with dotted perimeters indicate that the trip is cancelled.

Fig. 6a depicts the situation before rescheduling, with all three trips in the cut path cancelled. Of the depicted vehicles, V128, V110 and V147 are performing a trip at the moment VB breaks down and V13 is performing a deadhead. The other vehicles are waiting in the depot. In Fig. 6b, the schedule corresponding to the initial solution of the INE heuristic is presented. As can be seen, V165

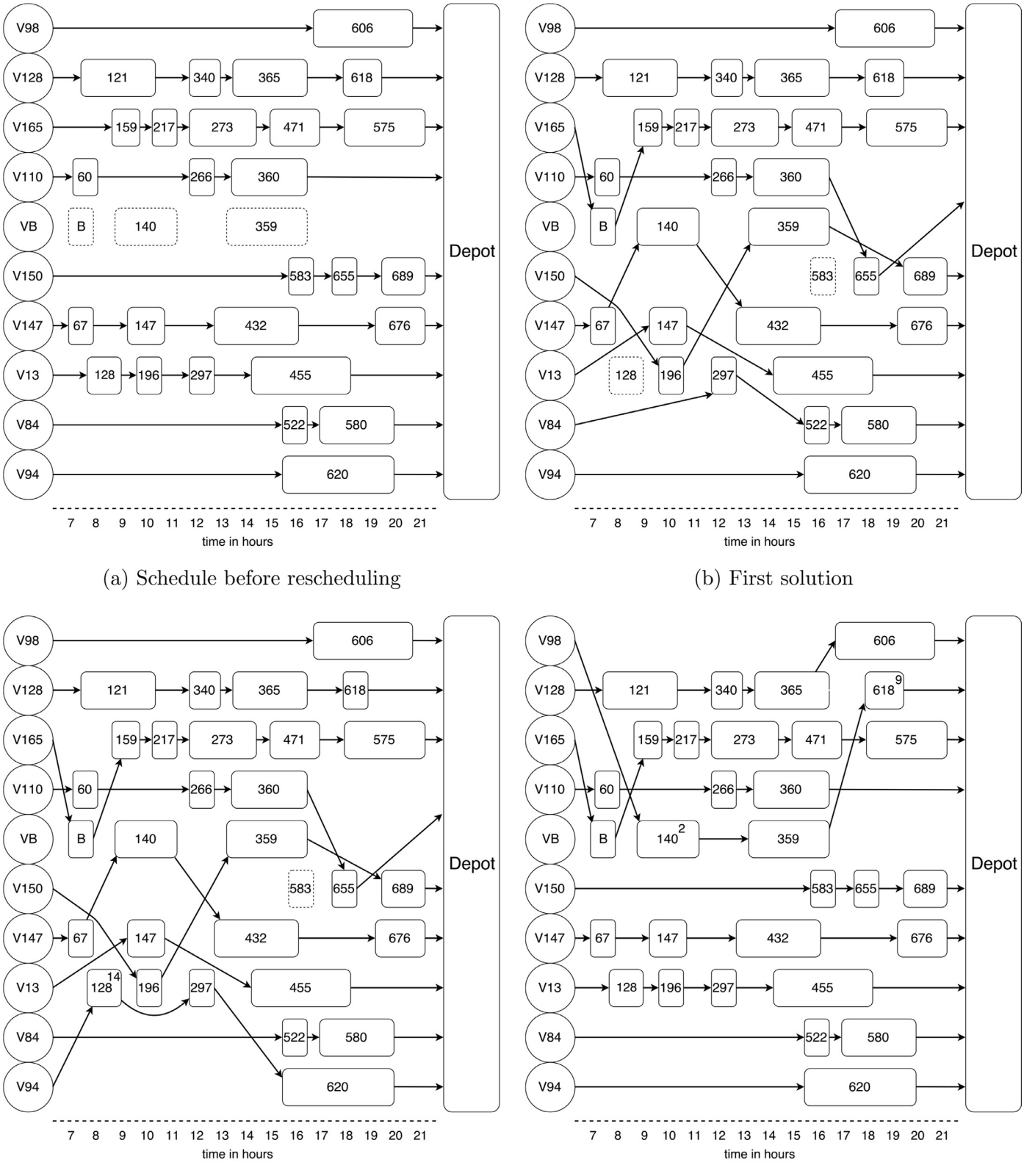

Fig. 6. Visualization of the initial schedule and solutions found by the INE heuristic. Only the schedule for the vehicles that are rescheduled somewhere in the process are depicted. A node with a dotted perimeter corresponds to a cancelled trip. If a trip is delayed, the superscript indicates the length of the delay.

Table 5
Results of the INE heuristic compared to the heuristic without retiming in class S, for different values of the reassignment penalty P and the delay costs per minute v.

		Without retiming			With retiming				
P	v	Cancelled trips	Reassigned trips	Costs	Cancelled trips	Reassigned trips	Delayed trips	Costs	Fewer cancellations due retiming (%)
50	10	0.92	6.14	2342	0.04	8.02	0.94	560	78
	20				0.10	7.82	0.88	732	74
500	10	0.92	5.00	4599	0.46	5.42	0.62	3826	40
	20				0.44	5.48	0.58	3845	40

Table 6
Characteristics of the solutions where the INE heuristic exploration heuristic was successful in class S, for different values of the reassignment penalty P and the delay costs per minute v.

P	v	Cancelled trips	Reassigned trips	Delayed trips	Average delay (min.)	Cost reduction (%)	$\|R\|$
50	10	0.00	8.84	1.24	7.38	82	50
	20	0.03	8.81	1.22	6.66	78	66
500	10	0.18	6.00	1.41	7.58	36	91
	20	0.18	6.05	1.32	7.90	34	90

performs the back-up trip. V147 and V150 perform the other trips in the cut path. As a consequence, trip 128 and trip 583 need to be cancelled. The cost of this solution is 4906.

The INE heuristic identifies 16 difficult trips in the first solution, among which trips 128, 196, 297, 583 and 655 are present in the figure. The difficult trips are allowed to be delayed in the second iteration of the INE heuristic. In the second iteration, a better solution is found within 15 s, which is presented in Fig. 6c. In this solution, trip 128 is delayed with 14 min. Trip 583 is however still cancelled. The cost of the solution is 2961.

In the second solution, no new difficult trips can be identified. Therefore, the neighborhoods of difficult trips are explored, leading to 11 trips that are added to $|R|$. In the next iteration, the INE heuristic again finds an improvement over the previous solution within 15 s. The final solution is visualized in Fig. 6d. It can be seen that by delaying trip 140 with 2 min and trip 618 with 9 min, all cancellations can be avoided. Even more, the final solution also contains substantially fewer reassignments compared to the first two solutions. The final solution has a total cost of 421.

The process described above is typical for the behavior of the INE heuristic. If the heuristic finds improvements over the initial solution without retiming, it finds these improvements quickly. Furthermore, if there are two or more cancelled trips in the initial solution, improvements are found in multiple steps.

7.3. Sensitivity analysis

In the previous section we demonstrated the effectiveness of the INE heuristic and we are able to conclude that the introduction of retiming leads to better solutions. In this section we repeat the analysis with different values for the reassignment penalty P and the delay costs per minute v. As we observed that the number of reassignments was very limited in the previous section, we test if we can reduce the number of cancellations even further if we set P at 50 instead of 500. Moreover, we also consider a delay costs v of 20 per minute instead of 10, as it is interesting to test whether the effectiveness of the heuristic deteriorates if we increase the delay costs.

In Table 5 we present the average number of cancelled trips, reassigned trips, delayed trips and the average costs of the solutions found in class S by the INE heuristic and by the heuristic without retiming for the different values of P and v. Table 6 shows the average number of cancelled, reassigned and delayed trips, the average delay length, the average cost reduction and the average $|R|$ of the solutions where cancellations were avoided using retiming in class S, for the different values of P and v. Tables 7 and 8 contain the same results for class M.

We start by discussing the results for class S. In Table 5 we observe that if P is 50 instead of 500, the average number of cancelled trips in the solutions without retiming is unaffected. On the other hand, the number of reassigned trips is higher (6.14 versus 5.00). Still, the number of reassignments is rather small considering that theoretically more than 500 trips could be reassigned. For the solutions of the INE heuristic (with retiming), the number of cancelled trips does reduce substantially when the reassignment penalty is lower. The INE heuristic finds improvements more often with the lower reassignment penalty, in 74–78% of the instances instead of in 40%. The increase of the delay costs per minute has a very limited effect if P is 500. If P is 50, the larger delay costs result an increase of the number of cancelled trips. In return, the average number of reassigned trips and delayed trips reduces. Overall, the fewest number of cancelled trips is 0.04 and is obtained using P equal to 50 and v to 10.

The characteristics of the solutions where retiming reduced the number of cancellations are in line with our expectations, see Table 6. If P is lower, fewer trips are cancelled and more trips are reassigned. The difference between the solutions obtained with the two delay costs is small. For the lower reassignment penalty, increasing the delay costs from 10 to 20 reduces both the average number of delayed trips and the average delay length. In contrast, for the higher reassignment penalty, the average delay length slightly increases. However, this increase is compensated by a larger decrease in the number of delayed trips, such that the total delay still decreases as one would expect.

The results for class M are similar to those of class S. In Table 7 we observe that the lower reassignment penalty results in fewer cancelled trips and more reassigned trips. Next to that, the INE heuristic finds improvements in a larger fraction of the cases if P is 50, which also causes more trips to be delayed compared to the higher reassignment penalty. Increasing the delay costs leads to more cancelled trips, although the difference is very small if P is 500 (0.80 versus 0.82). On the other hand, in Table 8 we observe that the higher delay costs lead to decreases of the number of delayed trips and the average delay for both values of the reassignment penalty.

From the sensitivity analysis we can conclude that the performance of the INE heuristic is quite robust with respect to the reassignment cost and delay costs per minute. For all values of the parameters, large improvements are found over solutions without retiming. Averaged over all parameter values and classes, the INE heuristic improves over solutions without retiming in 58% over the

Table 7
Results of the neighborhood exploration heuristic compared to the heuristic without retiming in class M, for different values of the reassignment penalty P and the delay costs per minute v.

		Without retiming			With retiming				
P	v	Cancelled trips	Reassigned trips	Costs	Cancelled trips	Reassigned trips	Delayed trips	Costs	Fewer cancellations due retiming (%)
50	10	1.38	6.80	3679	0.54	7.16	1.46	1834	66
	20				0.64	6.92	1.12	2103	64
500	10	1.44	3.86	5861	0.80	4.36	0.96	4596	54
	20				0.82	4.36	0.84	4675	50

Table 8
Characteristics of the solutions where the neighborhood exploration heuristic was successful in class M, for different values of the reassignment penalty P and the delay costs per minute v.

P	v	Cancelled trips	Reassigned trips	Delayed trips	Average delay (min.)	Cost reduction (%)	$\|R\|$
50	10	0.29	7.91	2.09	6.37	82	80
	20	0.44	7.59	1.65	5.91	73	84
500	10	0.66	5.10	1.66	8.04	36	85
	20	0.59	5.11	1.56	7.60	36	76

cases, reducing the average number of cancelled trips from 1.2 to 0.5. To achieve this reduction, on average 1 or 2 trips are delayed with 6–8 min.

8. Conclusion

In this paper, we discussed the vehicle rescheduling problem with retiming (VRSPRT). As the problem becomes intractable within reasonable time if we allow retiming for all trips, we proposed an iterative neighborhood exploration heuristic that allows retiming for a relevant subset of trips. Computational experiments indicate that the heuristic performs well. We find improvements over solutions that do not allow retiming in 58% of the tested instances. By delaying only one or two trips with on average 6–8 min, the average number of cancelled trips over all instances is reduced from about 1.2 to 0.5. This shows that by using retiming, it is possible to find better rescheduling solutions.

As we limited ourselves to vehicle rescheduling in this paper, a relevant topic for future research is to combine the vehicle rescheduling with crew rescheduling. This can range from varying the reassignment penalties to account for specific crew requirements up to complete integration.

References

Bunte, S., Kliewer, N., 2009. An overview on vehicle scheduling models. Public Transp. 1 (4), 299–317.

Carpaneto, G., Dell'Amico, M., Fischetti, M., Toth, P., 1989. A branch and bound algorithm for the multiple depot vehicle scheduling problem. Networks 19 (5), 531–548.

Cherkassky, B.V., Goldberg, A.V., Radzik, T., 1996. Shortest paths algorithms: theory and experimental evaluation. Math. Program. 73 (2), 129–174.

Dávid, B., Krész, M., 2017. The dynamic vehicle rescheduling problem. Central Eur. J. Oper. Res. 25 (4), 809–830.

Desaulniers, G., Lavigne, J., Soumis, F., 1998. Multi-depot vehicle scheduling problems with time windows and waiting costs. Eur. J. Oper. Res. 111 (3), 479–494.

Eggenberg, N., Salani, M., Bierlaire, M., 2010. Constraint-specific recovery network for solving airline recovery problems. Comput. Oper. Res. 37 (6), 1014–1026.

Held, M., Karp, R.M., 1971. The traveling-salesman problem and minimum spanning trees: part II. Math. Program. 1 (1), 6–25.

Huisman, D., Freling, R., Wagelmans, A.P., 2004. A robust solution approach to the dynamic vehicle scheduling problem. Transp. Sci. 38 (4), 447–458.

Huisman, D., Freling, R., Wagelmans, A.P., 2005. Multiple-depot integrated vehicle and crew scheduling. Transp. Sci. 39 (4), 491–502.

Kliewer, N., Amberg, B., Amberg, B., 2012. Multiple depot vehicle and crew scheduling with time windows for scheduled trips. Public Transp. 3 (3), 213–244.

Li, J.-Q., Borenstein, D., Mirchandani, P.B., 2008. Truck schedule recovery for solid waste collection in porto alegre, brazil. Int. Trans. Oper. Res. 15 (5), 565–582.

Li, J.-Q., Mirchandani, P.B., Borenstein, D., 2004. Parallel auction algorithm for bus rescheduling. In: Computer-Aided Systems in Public Transport. Springer, pp. 281–299.

Li, J.-Q., Mirchandani, P.B., Borenstein, D., 2009. A lagrangian heuristic for the real–Time vehicle rescheduling problem. Transp. Res. Part E 45 (3), 419–433.

Løve, M., Sørensen, K.R., Larsen, J., Clausen, J., 2002. Disruption management for an airline - rescheduling of aircraft. In: Applications of Evolutionary Computing. Springer, pp. 315–324.

Potthoff, D., Huisman, D., Desaulniers, G., 2010. Column generation with dynamic duty selection for railway crew rescheduling. Transp. Sci. 44 (4), 493–505.

Stojković, M., Soumis, F., 2001. An optimization model for the simultaneous operational flight pilot scheduling problem. Manage. Sci. 47 (9), 1290–1305.

Thengvall, B.G., Yu, G., Bard, J.F., 2001. Multiple fleet aircraft schedule recovery following hub closures. Transp. Res. Part A 35 (4), 289–308.

Veelenturf, L.P., Potthoff, D., Huisman, D., Kroon, L.G., 2012. Railway crew rescheduling with retiming. Transp. Res. Part C 20 (1), 95–110.

Visentini, M.S., Borenstein, D., Li, J.-Q., Mirchandani, P.B., 2014. Review of real-Time vehicle schedule recovery methods in transportation services. J. Schedul. 17 (6), 541–567.