Passenger-Aware Real-Time Planning of Short Turns to Reduce Delays in Public Transport

Julian Patzner
Martin-Luther-Universität Halle-Wittenberg, Germany

Ralf Rückert
Martin-Luther-Universität Halle-Wittenberg, Germany

Matthias Müller-Hannemann
Martin-Luther-Universität Halle-Wittenberg, Germany

Abstract

Delays and disruptions are commonplace in public transportation. An important tool to limit the impact of severely delayed vehicles is the use of short turns, where a planned trip is shortened in order to be able to resume the following trip in the opposite direction as close to the schedule as possible. Short turns have different effects on passengers: some suffer additional delays and have to reschedule their route, while others can benefit from them. Dispatchers therefore need decision support in order to use short turns only if the overall delay of all affected passengers is positively influenced. In this paper, we study the planning of short turns based on passenger flows. We propose a simulation framework which can be used to decide upon single short turns in real time. An experimental study with a scientific model (LinTim) of an entire public transport system for the German city of Stuttgart including busses, trams, and local trains shows that we can solve these problems on average within few milliseconds. Based on features of the current delay scenario and the passenger flow we use machine learning to classify cases where short turns are beneficial. Depending on how many features are used, we reach a correct classification rate of more than 93% (full feature set) and 90% (partial feature set) using random forests. Since precise passenger flows are often not available in urban public transportation, our machine learning approach has the great advantage of working with significantly less detailed passenger information.

2012 ACM Subject Classification Applied computing → Transportation

Keywords and phrases Public Transportation, Delays, Real-time Dispatching, Passenger Flows

Digital Object Identifier 10.4230/OASIcs.ATMOS.2022.13

Funding This work has been partially supported by DFG under grant MU 1482/7-2.

1 Introduction

Public transport is used by millions of people every day. It is of great importance for meeting mobility needs and achieving sustainability goals. However, its attractiveness suffers from disruptions and delays that increase travel times for passengers and lead to their frustration. The task of real-time dispatching is therefore to minimize the effects of disruptions. In general, there are a couple of control actions which can be used to mitigate delays.

In this paper, however, we will focus specifically on scenarios where single vehicles are excessively delayed. Such delays may be due to many kinds of problems, for example technical problems with the engine or the signaling system, temporarily blocked track or road sections, and the like. If such a heavily delayed vehicle is planned for subsequent (return) trips on the same line, the delay may propagate over quite some time. In such a scenario it might be advisable to consider the possibility of introducing a short turn to reduce the vehicle delay. Executing a short turn means that the current trip of the vehicle is prematurely terminated at some stop. Afterwards, the same vehicle reverses direction and resumes with the follow-up trip from this stop. That means, all intermediate stops till the terminus of the line are

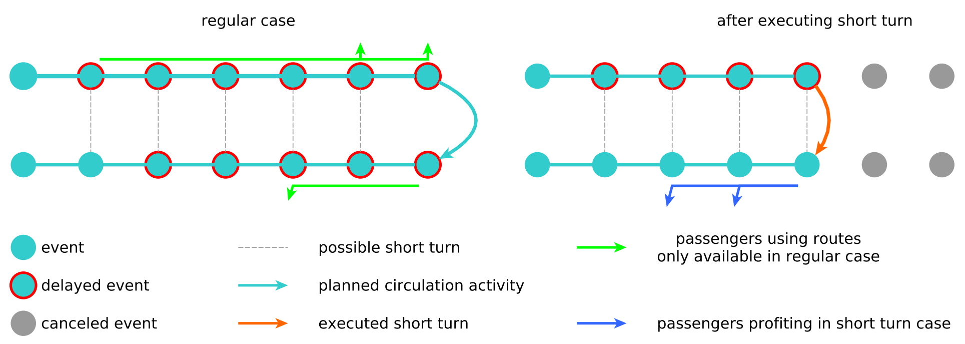

regular case
after executing short turn
event
delayed event
canceled event
possible short turn
planned circulation activity
executed short turn
passengers using routes only available in regular case
passengers profiting in short turn case

Figure 1 Sketch of both possibilities of a short turn scenario. A vehicle has a delay at its second stop. The left part shows the regular scenario without short turn. In this case, the delay is propagated to the following ten stops. On the right side is the scenario where a short turn is executed at the fifth stop of the line. Some passengers must use alternate routes to their target during a short turn (green) while other passengers (blue) benefit.

skipped in both directions. Ideally, the planned vehicle schedule can be restored in this way. Clearly, a short turn negatively affects all passengers currently using the vehicle, and those who wanted to board at some of the skipped stops. On the other hand, other passengers take advantage if the delay is reduced for subsequent trips. Figure 1 provides a systematic sketch of such a scenario. For a passenger-aware control strategy, dispatchers therefore need decision support in order to use short turns only if the overall delay of all affected passengers is reduced. Formally, we would like to support solving the following decision problem:

Short turn dispatching problem. Given a timetable T, a corresponding vehicle schedule S, a passenger flow pf, a delay scenario with delay predictions *delay* and a heavily delayed trip tr (i.e. with a delay exceeding some threshold parameter), decide whether there is a suitable opportunity to introduce a short turn in order to reduce overall vehicle and passenger delays (and select the one with best utility).

We consider the following general workflow (Figure 2): There is a central dispatching centre monitoring the delay scenario in real-time, i.e., the current delay status of every vehicle in the public transport system. Based on this delay information, a propagation model is used which predicts how the delay of each vehicle will evolve over time. Whenever some vehicle is heavily delayed, possible candidate stops for executing a short turn are identified and checked for their feasibility, i.e. technical requirements on the infrastructure like necessary switches for trains or turning options for busses have to be fulfilled. All potentially suitable candidates then have to be evaluated. In each case, the two alternatives of executing or not executing a short turn must be simulated with their consequences on all affected passengers. Finally, the best alternative is chosen.

Contribution. We propose a simulation framework for solving the short turn dispatching problem. An essential component of our framework are passenger flows. While high quality estimations of passenger flows based on ticket data are available in long-distance train traffic, data availability is much worse in local public transport for trams, busses, and local trains. More and more, automated fare collection data and automated passenger count systems can be used to infer passenger flows. However, it is still challenging to infer correct passenger

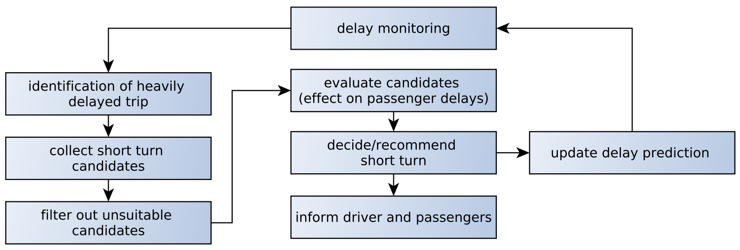

delay monitoring
identification of heavily delayed trip
collect short turn candidates
filter out unsuitable candidates
evaluate candidates (effect on passenger delays)
decide/recommend short turn
inform driver and passengers
update delay prediction

Figure 2 Steps in an online framework recommending short turns.

destinations from these data. We go a step further by assuming that sufficiently precise passenger flow data is available. That means, for each passenger we know exactly the intended route, i.e. the sequence of planned trips. In particular, we know the planned arrival time at the destination according to schedule. The main advantage of considering precise passenger routes is that we can take care of each individual passenger in case of disruptions. Thus, when solving the short turn dispatching problem, we can calculate the effect on the passenger's delay for both scenarios (executing a short turn or not). Moreover, based on the decision, we can recommend alternative routes that will get the passenger to the desired destination on a route minimizing a cost function, for example, reaching the destination as quickly as possible. Within our simulation, the bulk of the work is therefore the computation of optimal routes in an online scenario for all affected passengers. Though we have to solve many shortest path routing queries, we still achieve an excellent performance. In computational experiments with a public transport network for the German city of Stuttgart we can solve the short turn decision problem on average within few milliseconds, i.e. fast enough for real-time use.

We further analysed in which cases short turns are beneficial. We observe a clear correlation between beneficial short turns and multiple indicators that we can compute before the rerouting process of the simulation starts. Since our assumptions on the availability of precise passenger flows may be way too strong for urban public transportation, we try to relax them and work with significantly less detailed passenger information. Our second contribution uses machine learning (ML) techniques in order to classify cases where short turns are advantageous. To this end, we carefully selected a few features of the current delay scenario and the passenger flow to train an ML model. The best classification rate could be achieved with random forests. For the full feature set we obtain correct predictions in more than 93% of all cases. We also experimented with a smaller feature set using only partial information. Here we still get a success rate of more than 90%.

Related work. The importance of taking the passenger perspective into account has been stressed by Parbo et al. [18]. Our simulation framework for the real-time decision support of short turns is inspired by the PANDA framework for wait-depart decisions in delay management for long distance trains [14, 21]. The optimal capacity-aware rerouting of passengers after train cancellations has been considered in [13].

Ge et al. [4] provide a survey on disturbances in public transport and approaches to cope with them in planning and operations. Likewise, the review by Gkiotsalitis et al. [5] discusses several real-time control measures, including vehicle holding, stop-skipping, speed control, rescheduling, interlining, rerouting, and boarding limits. Wang et al. [26], for example, study the adjustment of dwell times and running trains on different speed levels as control actions.

The use of short-turning within real-time optimization has first been studied in Shen and Wilson [25]. They provide a non-linear mixed integer programming formulation that simultaneously tries to optimize holding times, stop skipping and short turns. They report experimental results with a single line (Boston, MA). Similarly, Nesheli et al. [16] combine mixed integer programming with a simulation framework. A case study for Auckland, New Zealand, based on three bus routes shows quite promising results. Our approach differs in several respects. First, Nesheli et al. allow short turns only if the vehicle has passed the last transfer point in the current direction. In contrast, we do not impose such a restriction. Second, Nesheli et al. assume that passengers disadvantaged by short turning may either walk to their intended destination or wait for the next vehicle on the line. We are more flexible by assuming that stranded passengers switch to an optimal alternative route to reach their final destination. This, of course, requires a much higher computational effort for finding these routes. Finally, our analysis is based on much larger networks. Liu et al. [10] study a short turn strategy for bus operations based on origin-destination (OD) data, but without taking actual delays into account. Note that short turning does not apply to each public transport line. In particular, ring or circle lines require other control actions [8].

To deal with both spatial and temporal peak periods of demand, the planning of short turns has also been considered as part of the strategic planning phase as a means to balance capacity and demand and so to mitigate crowding [1, 12].

Passenger flow prediction is an area of active research. For example, Kang et al. [6] study how to use automated fare collection data for generating OD matrices. Liu and Chen [9], Luo et al. [11], and Nagaraj et al. [15] present deep learning approaches for passenger flow prediction in bus transit systems. We therefore expect the quality of the available passenger flows for real-time dispatching to improve continuously.

Overview. The remainder of this paper is structured as follows. In Section 2, we describe in detail our simulation framework for the short turn dispatching problem. Results of our experimental study with the public transport network of the city of Stuttgart are presented in Section 3. Afterwards, in Section 4 we explain our machine learning approach for deciding upon short turns and present corresponding computational results. Finally, in Section 5, we conclude with a short summary and an outlook.

2 Short Turns During Daily Operation

In this section, we first provide the basic background for our simulation framework and then describe in detail how the short turn dispatching problem is solved.

2.1 Basic Model

Public transport infrastructure. The public transport infrastructure consists of stops and direct connections between them. These can be roads in bus transportation or the tracks in rail, tram or underground transportation. This network is usually called *public transportation network* (PTN). A *line* is a path in this network, i.e. a sequence of stops and direct connections between them.

Timetables and event-activity networks. A public transport timetable can be modeled as an event-activity network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ [24, 14] which contains nodes (*events* $e \in \mathcal{E}$) for every arrival or departure of a vehicle at a stop and directed edges (*activities* $a \in \mathcal{A}$) between them. For every major relationship between events there is a special type of activity. Every type of activity is directed and goes either from departure to arrival $dep \rightarrow arr$ or $arr \rightarrow dep$. The relevant types of activities are:

drive ($dep \rightarrow arr$) an activity of a vehicle driving from one stop to another,
wait ($arr \rightarrow dep$) a dwelling activity of a vehicle,

transfer ($arr \to dep$) an activity for passengers representing a possible interchange between vehicles (usually including a footpath),

turnaround ($arr \to dep$) an activity of a vehicle to reach its next trip.

A *trip* is a path of drive and wait activities that needs to be operated by a single vehicle. The *vehicle schedule* assigns to each trip a vehicle with a certain capacity, specifying the maximum number of passengers who can use it simultaneously. The vehicle schedule implies the turnaround activities in the event-activity-network.

Each event is equipped with several timestamps, specifying the planned time according to the timetable and the realized or estimated time with respect to the real situation. The time difference of events connected by activities yields the planned or actual duration for the activity. Under optimal conditions some activities may be performed faster. For activity a, the lower bound L_a specifies the minimum execution time of the activity. A delayed vehicle can reduce its delay if the scheduled duration of an activity is larger than its lower bound. The difference in time is called *slack* or *buffer*.

Delay propagation. For real-time dispatching decisions it is necessary to maintain an up-to-date and consistent internal model of the current delay scenario and estimated timestamps for future events. Hence, in a live system vehicles will regularly send their GPS coordinates. From these data, current status information and delays can be inferred. For our simulation framework, we can simulate such a stream of messages as well as artificial delay scenarios. Delays are propagated along the activities in the network. In the basic model, delays are transferred along drive, wait and turnaround activities, but available slack is used to reduce positive delays. It is assumed that vehicles never depart before their planned departure time. Delays may also be propagated along transfer activities if delay management uses waiting time rules between trips. More sophisticated propagation algorithms may also use the available free vehicle capacity and the number of boarding and leaving passengers to estimate actual dwell times and to model phenomena like bus bunching.

Passenger Model. After the event-activity network is constructed passenger flows are generated. For our networks there is an origin-destination matrix specifying demand. Every passenger has a starting location, target location, and a preferred time of departure. A *route* (or *journey*) of a passenger is a directed path in the event-activity network which starts at the starting location and has to end at the target. Before the online simulation is started passengers will select an initial route based on their demand and a utility function. For simplicity, the utility function is here taken as a weighted sum of the travel time and the number of transfers used, but can be customized within our framework. During the simulation we use the following model how passengers behave, depending on their knowledge about the current delay situation. In an online scenario source delays are revealed only step by step, namely soon after they have occurred. An exception to this are changes of the timetable caused by dispatching decisions that can be communicated once a decision is made. In our model, passengers are informed about delays and change their routes according to their utility function and certain rules:

1. Passengers are assumed to arrive at the very first stop of their route as planned. At that point of time, they check the validity of the planned route and choose an optimal route with respect to their utility function.
2. Once a passenger has entered a vehicle, he or she will stay on board of this trip till the planned exit stop where they either reach their final destination or transfer to another trip. That means, passengers are assumed not to change their route before they reach a natural decision point, usually a transfer stop.

3. If a planned transfer is missed, the passenger selects a new route subject to the current delay scenario.
4. If the current trip on the planned route is on time, a passenger will not switch to another route that has become possible because of delays of other trips.

Note that rule (2) and (4) may result in sub-optimal passenger routes but such a behavior seems natural for most passengers. One important aspect of this model is that in networks where good alternatives are available rule (1) results in scenarios where delayed vehicles are less occupied than initially planned.

2.2 Deciding Short Turns

Short turn model. Let $t_1 = (s_1, s_2, \dots, s_{n-1}, s_n)$ and $t_2 = (s_n, s_{n-1}, \dots, s_2, s_1)$ be two consecutive trips of the same vehicle, where the sequence of stops in t_2 appears in reversed order. A short turn st is a turnaround activity from an arrival event $from(st)$ of t_1 to a departure event $to(st)$ of t_2 at some non-terminal stops of t_1. Every activity between $from(st)$ and the end of t_1 and between the start of t_2 and $to(st)$ would be cancelled. A short turn is thus a premature turnaround edge that is not part of the planned schedule, but rather decided at short notice. Every passenger whose planned route includes a cancelled event has to be rerouted, while other passengers benefit from the reduced delay. The following sections explain how the (possible) benefit of a short turn is evaluated.

Detecting and filtering short turns. A short turn from an arrival event of trip t_1 to a departure event of trip t_2 will be considered as a candidate if the departure event of trip t_2 is sufficiently delayed, i.e. by more than a certain threshold δ, say, for example by at least 10 minutes. In order to recognize a potential short turn, we keep a list of trips exceeding this threshold with respect to the current delay status. Entries in this list are sorted increasingly by the time when the trip delay exceeds δ for the first time and processed in this order. The simulation proceeds in discrete time steps of one minute. The current time of the simulation is called *simulation time*. Some potential short turns are discarded before being evaluated. First of all, they have to be technically feasible. In addition, a short turn between two trips t_1 and t_2 is only relevant if no later short turn exists for which t_2 can depart according to schedule. In other words, the first relevant short turn candidate st_f is the latest short turn possibility for which t_2 can depart according to schedule. Short turns at earlier stops of trip t_1 would be suboptimal, since more stops have to be cancelled and more passengers rerouted. Later short turns, however, can not be prematurely excluded: while t_2 would not be able to depart according to schedule, they are less disruptive for the passengers and may be the preferable option, especially if a large number of passengers want to enter or exit at a later stop.

Note that due to the definition of a circulation edge all simulations of a short turn meet requirements of the vehicle schedule (rolling stock), but do not contain any information about the crew schedule. In practice, a short turn can not occur when necessary crew changes are skipped.

Computation of scenarios. For every trip in the list of heavily delayed trips, the first relevant short turn st_f to this trip is calculated or updated after a delay has been recorded and propagated. The event $from(st_f)$ is kept in a simple array of lists mapping the simulation time to these departure events. This array is checked when the simulation time is updated. For every event in the list corresponding to the current time, the following “decision” subroutine is called. This subroutine, formally described in Algorithm 1 (see Appendix A), recommends whether trip t should execute a short turn. Since multiple short turns are in competition, all of them have to be evaluated. Starting with the stop of the first relevant short turn st_f, we evaluate the short turn at each stop of the trip. As a cost measure for a scenario we use the total arrival delay of all passengers in the network, including a penalty of 300 seconds for each transfer. Alternatively and without extra effort, we could also use the sum of the squared arrival delays. To find the short turn with the largest reduction of the cost function, an evaluation subroutine is called for every possible short turn of t, starting with $st_f(t)$. This second subroutine will be explained in the following paragraph. New routes are calculated for the affected passengers of a short turn. The decision subroutine saves these routes for the best currently found short turn. At the end of the decision subroutine, the best found short turn will be executed if the reduction of the cost function is larger than or equal to some threshold θ_{thr}. The routes of the affected passengers and the changed times of the affected trips are updated. For the efficient computation of optimal passenger routes, we use a modified version of the connection scan algorithm (CSA)[2]. The algorithm has been modified in order to calculate routes based on composite cost functions.

Evaluation of scenarios with distinct types of affected passengers. The evaluation subroutine, described in Algorithm 2 (for a detailed pseudocode see Appendix A), calculates the difference of cost function for executing and not executing the specific short turn st, based on the state of the network at the time the decision subroutine is called. First, we determine the sets of passengers affected by the potential short turn st. We differentiate four different types of passengers. Table 4 in Appendix A provides a compact reference describing these types and how they can be determined.

The first type contains passengers which would have part of their current route canceled if the short turn is executed. Finding these passenger groups is simple: we trace the activity edges of the current vehicle from $from(st)$ to $to(st)$ and put every passenger on these edges in a set P_{-A}. These passengers would have to be rerouted if the short turn is executed. It is also possible that these passengers have a broken transfer between $from(st)$ and $to(st)$, in which case a rerouting is required in both scenarios. The second type of passenger groups are passengers whose routes are only possible because the current trip is delayed. This is only possible for passengers who searched for their routes after the delay was announced. For these groups the short turn might be detrimental: a transfer into a vehicle with improved punctuality might become impossible if the transfer slack is too small. To find these passengers we trace the vehicle starting with $to(st)$. For every scanned event, we calculate the time improvement of this event resulting from the short turn. We then iterate over the transfer edges into the current event. If the slack of the transfer is zero or positive and smaller than the time improvement at the event, we add the corresponding passengers to a set P_{-B}.

The third type of passengers are those who currently have a broken transfer because of a delay, but the broken transfer becomes possible again if the short turn is executed. Finding these passengers is similar to finding the second type: we again trace the vehicle starting with $to(st)$ and calculate the time improvement at the current event. We then iterate over the transfer edges out of the event. We only consider transfer edges of passengers which are not in P_{-A} or P_{-B}. If the slack of the transfer edge is negative, meaning the transfer is not possible in the current state of the network, we add the passengers corresponding to the transfer edge to a set P_{+C} if the absolute value of the transfer slack is smaller than or equal to the time improvement of the current event. This is the case if the transfer is possible if short turn is executed. These passengers would have to be rerouted if the short turn is not executed.

Parallel to this, we consider the fourth type P_{+D} of affected passengers: passengers whose final arrival gets directly improved because of the short turn. This is the case if an event is the final destination of a passenger. In this case, we directly add the time improvement to a variable `direct_cost_improvements`. If a trip is delayed, passengers may search for a better alternative route. In case the short turn is executed, a better alternative route may still exist if the short turn doesn't remove the entire delay. To calculate the direct improvement, we first find the best alternative route according to current circumstances. We then compare these alternative costs to the costs of the current route for both scenarios and save the minimum of the two routes. For the scenario of executing the short turn, the time improvement at the final arrival is subtracted from the costs of the current route. The direct improvement of the short turn for the current passenger is the difference between these two minima. This is described in Algorithm 2 (Appendix A).

Computing the benefit. In order to calculate the total difference of the cost function, we compare both scenarios for each passenger found in the previous step. First, we calculate the cost for every group for the case that the short turn is not executed. For this case, we have to reroute the passengers in P_{+C} and the passengers in P_{-A} with broken transfers. For the other passengers, we take the costs of their current routes. Only the remaining costs of the routes are taken into account, starting at the current simulation time. After this step, we undo the reroutings and execute the short turn. The skipped edges are marked as canceled and the times of the subsequent trips are updated. In this scenario, P_{-A} and P_{-B} have to be rerouted. The calculated routes for P_{-A} are saved and returned. These will be applied in case the current short turn is the best short turn of the decision subroutine. The new routes for the other types of passenger groups are not saved because they lie in the future and will be rerouted in a different subroutine of the simulation. Before returning the total improvement of the network costs and the new routes, the network is reset.

3 Experiments

In this section, we describe our computational evaluation of the simulation framework for short turns.

Experimental setup. The simulation is written in C++ and compiled with gcc 9.4.0, using full compiler optimization, on Ubuntu 20.04. The experiments are run on an AMD Ryzen 7 5800X, clocked at 4.7 GHz during program execution, and 32 GB of 3600 MHz DDR4 RAM.

Public transportation network. We use realistic data for the public transportation network of Stuttgart, provided in the LinTim format [22, 23], including passenger demand. The Stuttgart instance [3] is a mixed network, consisting of train and bus lines. The network has 769 stops (or stations) and a passenger demand of 54 626 passengers per hour. During peak traffic 780 vehicles start their trips per hour across 111 unique used lines. Their maximum capacities reach from 70 passengers for busses up to 400 passengers for trams and up to 1000 for regional trains. Of the 780 trips per hour 556 are busses and 224 trains/trams. A specific timetable and vehicle schedule with respect to the given passenger demand has been optimized by the LinTim software.

Experiments 1 (single artificial delays). In order to evaluate the usefulness of short turns, we have to generate delay scenarios. In the first experiment, we generate an artificial starting delay for a single vehicle in the network. The delay is propagated along the trips of the vehicle. Each time the vehicle has a planned buffer time, the propagated delay is reduced by that amount. We delay only one vehicle because we want to guarantee that passengers are not influenced by other changes in the network unrelated to the evaluated short turn. We simulate four hours of passenger travel. This simulation is repeated for each trip in the network.

In order to capture how profitable a short turn is for a certain delay, we simulate a delay in the range of 10 to 30 minutes, in steps of five. We only consider delays that do not result in the delayed trip overtaking the next scheduled trip of the same line. We evaluate possible short turns as described in Section 2. Because this experiment only features independent delays, the resulting short turn recommendations guaranteed an optimum for the system.

For this and the following experiment, the penalty for a transfer was set to 300 seconds. We assume a short turn takes 180 seconds to execute. In total, we simulated 6786 vehicle delays for Stuttgart. We found 3343 short turns, 2232 of which reduce the accumulated cost function when the acceptance threshold θ_{thr} is set to 0.

Experiment 2 (more realistic random delays). While the first experiment is deterministic and has only one isolated source delay, the second experiment generates more realistic scenarios featuring multiple simultaneous, but independent delays. We would like to point out that our assumption of independent delays is clearly a simplification. In reality, there are frequent cases where delays are correlated because of an underlying reason. Our simulation framework would also work for more sophisticated delay scenarios. In this experiment, we simulate four hours of traffic. Each trip receives a random start delay according to a discrete delay distribution within a range of 0 to 10 minutes. Likewise, each driving activity receives an extra delay with respect to a second delay distribution. Figure 6 shows the cumulative distribution functions we used. The function is chosen so that 82% of starting events are not delayed, and 97.1% of driving activities do not get an additional source delay. With these probabilities, we used 10 independent runs to create a bigger data set of possible short turns than in the deterministic experiment. The number of short turn candidates generated was 33,159.

In contrast to the first experiment, there is no guarantee that a recommended short turn produces an optimal solution for the system. This is inherent to all online systems where future delays are not known beforehand. Moreover, we ignore complex cases where several heavily delayed vehicles should be considered in combination.

Results. General information about Experiment 1 is presented in Table 1. Note that delay here means the delay of a vehicle at the start of the simulations. The actual delay at the time of the short turn may be lower because of planned buffer times in the network. Note that more than one trip might be evaluated in one simulation. If the starting delay is high, it may propagate along multiple trips of a vehicle. The number of candidates decreases with higher starting delays because we only consider delays that do not result in the delayed trip overtaking the next scheduled trip of the same line.

It is expected that the average improvement increases as the starting delays increase. The average improvement and the percentage of trips with positive short turns is rather high for all starting delays. This can be explained by the fact that many trips are almost empty near to the end of the line, if the terminal stop is not a hub for transfers to other lines.

For simulations with ten minutes of starting delay, the average number of passengers on activities that would be cancelled by a short turn (with positive or negative improvement) is 70.5, but the median is only 14. For 22.2% of all trips with short turns, the cancelled

Table 1 For different starting delays (in minutes), this table presents how many trips were delayed (#trips), how many short turns were evaluated (#*st* evaluated), how many delayed trips with short turn candidates exist (#trips with *st* candidates), how many of them are beneficial (#beneficial *st*), and the average total improvement of costs for all passengers (avg improvement). The last two columns only consider the best short turn of a trip.

delay	#trips	#*st* evaluated	#trips with *st* candidates	#beneficial *st*	avg improvement
10	2518	1999	1214	784	152m 36s
15	1780	1856	801	536	187m 0s
20	1411	1309	481	312	187m 43s
25	1609	1868	574	405	336m 55s
30	954	982	273	195	464m 56s

activities were completely empty. One explanation for such a high number is that during morning peak hours, most passengers travel to the city centre, while outbound trips in the suburbs are not heavily loaded near to their terminus stop. Another main reason for so many empty trip segments is that whenever a vehicle has a substantial delay and other lines drive in the same direction, affected passengers take an alternative before the short turn is considered.

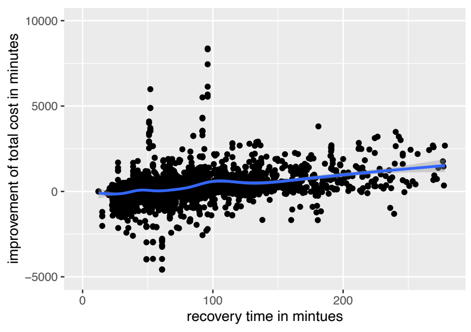
The time-critical operation is the finding of new routes for passengers. Across both experiments, each evaluation of a short turn takes 323.81 CSA queries on average, while the average total runtime for a short turn evaluation is 143.25 ms. The average runtime for each CSA query is 0.44 ms. This would enable a quick real-time evaluation of short turns, but in praxis, the limiting factor is usually the available data.

Features correlating with short turn recommendations. Figures 3a and 3b show how features of the scenarios correlate with the improvement in passenger utility if the short turn is executed. The *recovery time* of a delayed vehicle is defined as the time it would take the vehicle to completely eliminate its delay by utilizing buffer times within the timetable if no short turn is performed. Figure 3a plots the recovery time against the benefit in minutes, while in Figure 3b the known feature is the remaining delay of the vehicle at the stop before a short turn is executed. Both figures show a certain correlation between the value of the feature and the likelihood that the resulting short turn is beneficial. Figure 4a shows how the difference in the number of passengers belonging to groups that will benefit from a short turn and the non-beneficiaries correlating to the improvement in costs. Figure 4b shows that the difference in the number of reroutings is also correlated with the improvement. When multiple features indicate the affiliation of a scenario to the classes “recommend a short turn” and “do not recommend a short turn” we may not need further expensive computational effort to decide.

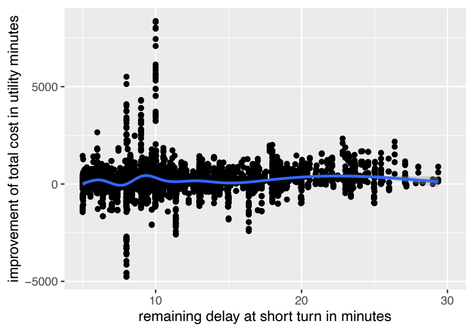
Further Evaluations. Figure 5 shows the fraction of beneficial short turns depending on the threshold imposed for accepting a short turn. Since dissatisfaction of passengers who have to suffer from short turns might outweigh the gain in punctuality for others, a threshold above zero seems preferable in practice.

4 Machine Learning Short Turn Decisions

For our simulations, we assumed perfect knowledge of the passenger flow. This is unrealistic for a practical scenario and we thus explore methods for the prediction of beneficial short turns with less information. In particular, we no longer want to assume that we know

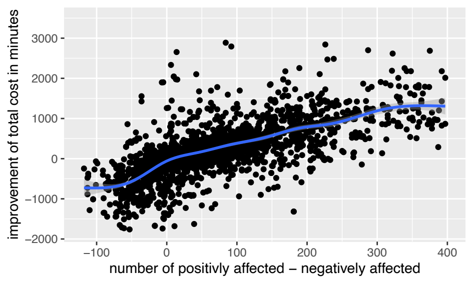


(a) Correlation of the duration it takes to recover the initial delay to improvement in costs.

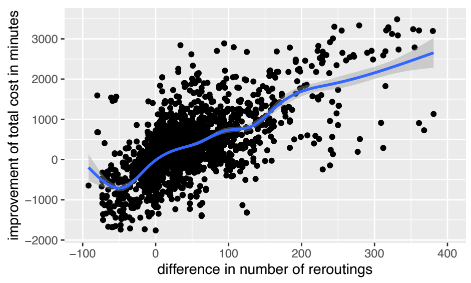


(b) Correlation between the delay at short turn and improvement. Aligned points belong to trips without buffer before the short turn.

Figure 3 Graphs showing basic information of scenarios correlating with benefit of improvement.

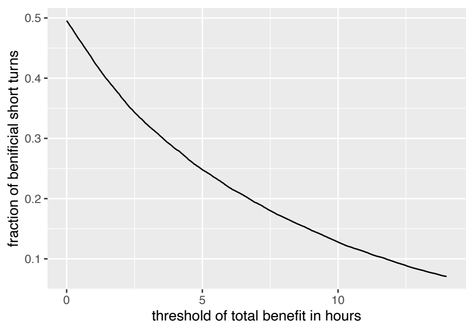


(a) The number of positively affected passengers minus the negatively affected passengers.

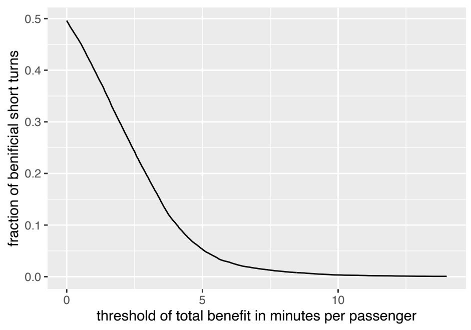


(b) Difference in reroutings.

Figure 4 Graphs showing basic basic information of scenarios correlating with benefit of improvement of a short turn.



(a) The fraction of beneficial short turns when there is a threshold of gained total passengers utility in hours. When the benefit for passengers has to be greater than a total of 5h only 25% of short turns are executed.



(b) The fraction of beneficial short turns when there is a threshold of gained average passengers utility of all passengers involved. When the benefit for all passengers involved has to be greater 5 minutes per passengers only 5% of short turns are executed.

Figure 5 In our definition a short turn is beneficial if there is any improvement in passenger utility. When dispatchers have a threshold for the minimum gained utility to execute a short turn the fraction of beneficial short turn declines sharply.

passengers' destinations. When we have easily computable features as input and their simulated recommendation as output we can apply methods from supervised machine learning to generate a model that predicts the recommendation without the computation of the scenario. Here we continue our work with the data sets from the previous section to see whether we can make predictions with a sufficiently high accuracy.

Defining features. The first step in this process of defining and producing features during the regular analysis of a short turn scenario is the creation of a vector of those features for all scenarios of our study. We selected 16 features that could be beneficial for machine learning algorithms to predict whether a short turn should be executed. Features that might be useful for the ML algorithms are the recovery time, the delay without the short turn, the number of passengers on canceled events, and the number of passengers on events that will have a smaller delay once the short turn is executed. For the machine learning of short turns, we use the following set of features:

1. time difference until delay is recovered because of buffer times
2. current time of day
3. current delay of the vehicle when short turn is considered
4. time saved if short turn is executed
5. number of passengers inside vehicle when short turn is considered
6. planned number of passengers in the subsequent reversed trip at the current stop if short turn is not executed
7. planned number of passengers in the subsequent reversed trip at the current stop if short turn is executed
8. number of remaining stops of the trip until the regular turnaround edge
9. predicted number of delayed trips of the current vehicle
10. difference in number of rerouting operations when simulating both scenarios
11. number of passengers in $P_{-A} \cup P_{-B}$ with valid routes before short turn
12. number of passengers in $P_{-A} \cup P_{-B}$ with invalid routes before short turn
13. number of passengers in P_{+C}
14. number of passengers in P_{+D}
15. number of (unique) passengers on the subsequent reversed trip until delay is recovered because of buffer times
16. number of (unique) passengers on the current vehicle until delay is recovered because of buffer times (can be higher than the previous feature if multiple trips are affected by the delay)

Note that the destination of passengers is not included in this data. This has the benefit that when dispatchers have knowledge of the number of passengers inside their vehicles but not their destinations, the predictions will still produce reliable results. The assumption that we know everything about the passenger flow becomes less important from this point forward. The set of 16 features is still more than the data a typical dispatcher would have at their disposal.

To this end, we produced a restricted set of features that a dispatcher might generate with little effort. These features are (1) the current delay of the vehicle before the short turn is executed, (2) the duration how long it would take to recover the delay by only consuming planned buffer times, (3) the number of minutes the delay is reduced by the short turn, (4) how many stops until the trip is over, (5) how many passengers are inside the vehicle when executing the short turn and (6) the estimated passenger demand of stops where the delay is decreased by the short turn.

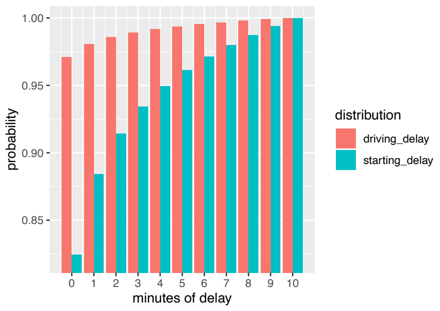


Figure 6 The cumulative distribution function for starting delay and driving delay function. The y-axis shows the probability that an event gets a delay equal to or smaller than the number of minutes at the x-axis.

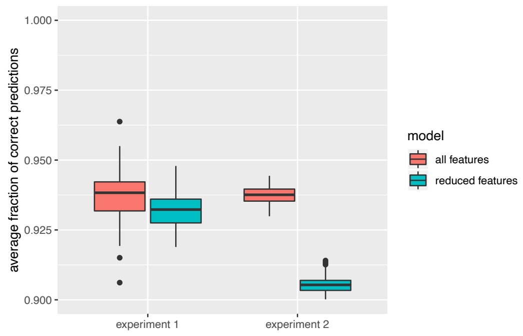


Figure 7 Comparison of Experiment 1 and 2: The average rate of correct predictions of a short turn recommendation with full and reduced feature sets.

Choosing platform and model. The process of creating a machine learning model has become easier during the last few years. Once the input (features) and output (recommendations) are stored in comma separate value files, a simple python script that uses a small number of open-source libraries can create good classifications. For our purposes we only needed NumPy [17] and the library *scikit-learn* [19]. There are many methods for creating a machine learning model for the purpose of classification of the input vector in only two classes. Nearest neighbor classification, support vector machines, and decision trees are examples of well-known methods. We tested ten different methods from the *scikit-learn* library [20] for creating a model and selected the model with the highest number of correct classifications within 100 trails where we split our data into 80 percent training- and 20 percent testing data. The data we used are all short turns simulated in Experiment 1 and 2.

The method that produced the best prediction on our data was the *RandomForestClassifier* (93.73% correct classifications on average) of *scikit-learn*. This evaluation includes parameter tuning as an alternative to using default parameters. However, the best parameters only improved the quality of estimation by an insignificant margin of .2%

Other relatively good methods were *DecisionTreeClassifier* (2% gap to *RandomForestClassifier*) and the *AdaBoostClassifier* (4% gap to *RandomForestClassifier*). All other tested methods did not consistently produce a classification rate above 90%.

Evaluating models for Experiment 1. To have sufficient data for the machine learning process we combined experiments with different initial delays to one data set. We created two models 'all features' and 'partial features' and trained the *RandomForestClassifier* with the training data. It is important to consider how many scenarios in the training data belong to the execute and do not execute-case. A classification data where 99% of all entries belong to one class must have a different prediction quality (usually much higher than 99%) than a classifier for data that has 50% of the training data belonging to one class. In our experiments, we test with different initial delays from 10 to 30 minutes. In these scenarios, short turns are beneficial in 66.52% of all cases. As mentioned earlier a short turn is beneficial as soon as there is one second of utility gained from the short turn. In practice, a short turn may only be executed if there is a sufficient gain (above some thresholds) in utility. The fraction of beneficial short turns drops significantly when a threshold is applied (see Figure 5).

The median prediction quality of the combined model with "all features" is 93.73%. This means that the process of predicting the right outcome of a short turn simulation can be predicted by machine learning if detailed data about the situation, passengers and

Table 2 Distribution of recommended short turn execution by simulation and classification by random forest classifier.

Experiment 1	simulation execute	simulation no execution
prediction execute	63.75%	3.50%
prediction no execution	2.77%	29.98%

Table 3 Distribution of recommended short turn execution by simulation and classification by random forest classifier.

Experiment 2	simulation execute	simulation no execution
prediction execute	47.95%	2.61%
prediction no execution	3.64%	45.80%

their routes are available. In comparison, the median prediction quality of the combined model with ‘‚†partial features’ is 93.22%. This means predicting the right outcome is still relatively good, even when the amount of information is only limited to a correct delay propagation/estimation, the number of counted passengers in the vehicle and knowledge about the demand of the next line. Table 2 shows the confusion matrix for this classification. We have 63.75% true positives and 29.98% true negatives, and only 3.50% false positives and 2.77% false negatives. The conclusion of this experiment is that for systems where delays are rare, short turns are often beneficial and this fact can be predicted with high accuracy.

We would like to point out that our classifiers can only predict whether a short turn candidate is beneficial, but not select the optimal short turn if several options exist. Choosing the optimal short turn candidate is much harder for machine learning since it has to predict the benefit using regression, and the essential comparison for a recommendation concerns cases that share a significant number of features. The first attempt for this method produced a prediction that could only determine the best of multiple beneficial short turns in 75.8% of all cases using the *XGBRegressor* from *scikit-learn*.

Evaluating models for Experiment 2. Before looking at the results of Experiment 2 we want to point out that it is harder to estimate the correct outcome than in Experiment 1 for one particular reason. In Experiment 1 we had more large isolated delays which made short turns more likely, namely in more than 66.52% of cases short turns were beneficial. Because of the buffer included in the schedule, a fraction of the delays is made up when the vehicle reaches stops where a short turn is possible. This can be seen in Figure 3b, where delays are very often less than the initial 10 to 30 minutes. Experiment 2 has more mid-sized delays (in the range $5 < x < 15$), but the delays are generated anywhere during the journey. This leads to a similar distribution of delays at possible short turns. The fraction of cases where short turns are beneficial is 51.59% and lower than in Experiment 1. The median prediction quality for the model with all features was 93.75% and thus as good as in Experiment 1 where delays were deterministic and isolated. Table 3 shows the confusion matrix for this classification. With reduced features, the estimation quality drops to 90.50%. So even in more realistic delay scenarios, the estimator predicts the correct short turn strategy with good accuracy, when all features are used. The reduced features produce an estimation such that it is likely that only 9 of 10 cases receive the correct prediction. For practical purposes, this might not be enough for automation, but it is certainly helpful for a first indication of whether a short turn is highly likely or unlikely to benefit passengers.

5 Conclusions and Outlook

We have presented a simulation framework for decision support of dispatchers in public transport focusing on the planning of short turns for heavily delayed vehicles. Our experiments have been conducted on a mid-size regional public transport network. Computation times turned out to be in the range of few milliseconds to decide whether a short turn would be beneficial and for computing alternative routes for all passengers affected by such an operation. For larger public transport systems and more dense passenger flows we expect that our approach will scale quite well. This is due to the fact that vehicle capacities are bounded and that the computational effort is more or less proportional to the number of rerouting queries which have to be done. A similar study could also be carried out for long-distance traffic. Again, we are confident that our findings can be transferred.

A reasonable extension of our model would be to include within the delay propagation that dwell times depend on the number of boarding and alighting passengers. Our current paradigm is that given circulation edges are binding because of a strict vehicle schedule. Without this restriction, a short turn could also feature more complex scenarios where a follow-up trip or other available vehicle resume the delayed future trip instead. More precisely, instead of short-turning the heavily delayed vehicle, one could consider short-turning the next vehicle on the line. The advantage could be that the first delayed vehicle is likely to be crowded, while the following vehicle might carry by far less passengers. Such an operation is slightly more complicated, but could be incorporated into our simulation framework, too. However, it should be noted that these operations have a larger impact on the staff schedule.

For machine learning models for detailed heterogeneous networks with many different types of vehicles and corresponding costs, capacities and possibilities for short turns, the feature sets should also include those. The current machine learning model only separates trains/tram and buses implicitly by the capacity of the vehicles.

Since we can make real-time decision support for waiting decisions and short turns it would be interesting to include stop skipping and dispatching additional vehicles when delays during peak traffic cause major problems concerning vehicle capacities. Supporting all of these control actions jointly is also an intriguing challenge. The prediction of favorable short turns succeeds satisfyingly well with the help of machine learning even if we exploit only simple countable or previously collected passenger data. As information on passenger flows and current delays is in practice always noisy and partially incomplete, the sensitivity of short turn recommendations is worth studying in more detail in the future. A similar analysis has been done for wait-depart decisions in PANDA[7]. Moreover, we could explore further which features are required to have a robust classifier for recommending short turns.

References

1 C. E. Cortés, S. Jara-Díaz, and A. Tirachini. Integrating short turning and deadheading in the optimization of transit services. *Transportation Research Part A: Policy and Practice*, 45(5):419–434, 2011. `doi:10.1016/j.tra.2011.02.002`.

2 J. Dibbelt, T. Pajor, B. Strasser, and D. Wagner. Connection scan algorithm. *Journal of Experimental Algorithmics*, 23, March 2017. `doi:10.1145/3274661`.

3 Collection of open source public transport networks by DFG Research Unit "FOR 2083: Integrated Planning For Public Transportation", 2018. URL: `https://github.com/FOR2083/PublicTransportNetworks`.

4 L. Ge, S. Voß, and L. Xie. Robustness and disturbances in public transport. *Public Transport*, 2022. `doi:10.1007/s12469-022-00301-8`.

5 K. Gkiotsalitis, O. Cats, and T. Liu. A review of public transport transfer synchronisation at the real-time control phase. *Transport Reviews*, 2022. `doi:10.1080/01441647.2022.2035014`.

6 M. J. Kang, S. Ataeian, and S. M. Mahdi Amiripour. A procedure for public transit OD matrix generation using smart card transaction data. *Public Transport*, 13:81–100, 2021. `doi:10.1007/s12469-020-00257-7`.

7 M. Lemnian, M. Müller-Hannemann, and R. Rückert. Sensitivity analysis and coupled decisions in passenger flow-based train dispatching. In M. Goerigk and R. Werneck, editors, *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*, volume 54 of *OpenAccess Series in Informatics (OASIcs)*, pages 2:1–2:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/OASIcs.ATMOS.2016.2`.

8 C. Liebchen, S. Dutsch, S. Jin, N. Tomii, and Y. Wang. The ring never relieves – response rules for metro circle lines. *Journal of Rail Transport Planning & Management*, 23:100331, 2022. `doi:10.1016/j.jrtpm.2022.100331`.

9 L. Liu and R.-C. Chen. A novel passenger flow prediction model using deep learning methods. *Transportation Research Part C: Emerging Technologies*, 84:74–91, 2017. `doi:10.1016/j.trc.2017.08.001`.

10 R. Liu, H. Yu, P. Wang, and H. Yan. A short-turn dispatching strategy to improve the reliability of bus operation. *Journal of Advanced Transportation*, Article ID 5947802, 2020. `doi:10.1155/2020/5947802`.

11 D. Luo, D. Zhao, Q. Ke, X. You, L. Liu, D. Zhang, H. Ma, and X. Zuo. Fine-grained service-level passenger flow prediction for bus transit systems based on multitask deep learning. *Trans. Intell. Transport. Sys.*, 22(11):7184–7199, 2021. `doi:10.1109/TITS.2020.3002772`.

12 S. Moon, S.-H. Cho, and D.-K. Kim. Designing multiple short-turn routes to mitigate the crowding on a bus network. *Transportation Research Record*, 2675(11):23–33, 2021. `doi:10.1177/03611981211003899`.

13 M. Müller-Hannemann, R. Rückert, and S. S. Schmidt. Vehicle capacity-aware rerouting of passengers in delay management. In V. Cacchiani and A. Marchetti-Spaccamela, editors, *19th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019)*, volume 75 of *OpenAccess Series in Informatics (OASIcs)*, pages 7:1–7:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/OASIcs.ATMOS.2019.7`.

14 M. Müller-Hannemann and R. Rückert. Dynamic event-activity networks in public transportation — timetable information and delay management. *Datenbank-Spektrum*, 17:131–137, 2017. `doi:10.1007/s13222-017-0252-y`.

15 N. Nagaraj, H. L. Gururaj, B. H. Swathi, and Y.-C. Hu. Passenger flow prediction in bus transportation system using deep learning. *Multimedia Tools Appl.*, 81(9):12519–12542, 2022. `doi:10.1007/s11042-022-12306-3`.

16 M. M. Nesheli, A. Ceder, and T. Liu. A robust, tactic-based, real-time framework for public-transport transfer synchronization. *Transportation Research Procedia*, 9:246–268, 2015. Papers selected for Poster Sessions at the 21st International Symposium on Transportation and Traffic Theory Kobe, Japan, 5-7 August, 2015. `doi:10.1016/j.trpro.2015.07.014`.

17 T. Oliphant. Numpy - the fundamental package for scientific computing with python, 2016. URL: `https://numpy.org/`.

18 J. Parbo, O. Nielsen, and C. Prato. Passenger perspectives in railway timetabling: a literature review. *Transport Reviews*, 36(4):500–526, 2016.

19 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

20 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Classifier comparison, 2022. URL: `https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html`.

21 R. Rückert, M. Lemnian, C. Blendinger, S. Rechner, and M. Müller-Hannemann. PANDA: a software tool for improved train dispatching with focus on passenger flows. *Public Transport*, 9(1):307–324, 2017.

22 A. Schiewe, S. Albert, P. Schiewe, A. Schöbel, and F. Spühler. LinTim - Integrated Optimization in Public Transportation. Homepage. `https://lintim.net`, 2020.

23 A. Schiewe, S. Albert, P. Schiewe, A. Schöbel, and F. Spühler. LinTim: An integrated environment for mathematical public transport optimization. Documentation for version 2020.12. Technical report, TU Kaiserslautern, 2020. URL: `https://nbn-resolving.org/urn:nbn:de:hbz:386-kluedo-62025`.

24 P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematic*, 2:550–581, 1989.

25 S. Shen and N.H.M. Wilson. *An Optimal Integrated Real-time Disruption Control Model for Rail Transit Systems*, pages 335–363. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. `doi:10.1007/978-3-642-56423-9_19`.

26 Y. Wang, M. Zhang, S. Su, T. Tang, B. Ning, and L. Chen. An operation level based train regulation model for a metro line. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2920–2925, 2019. `doi:10.1109/ITSC.2019.8916956`.

A Algorithms

Algorithm 1 Decision Subroutine.

```
input : first possible short turn st_f of trip current_trip
best_st ← nil;
(best_cost_improvement, new_routes) ← (0, nil);
for shortturn st of current_trip starting with st_f do
    (cost_improvement, temp_new_routes) ← evaluate(st);
    if cost_improvement > best_cost_improvement then
        best_st ← st;
        (best_cost_improvement, new_routes) ← (cost_improvement,
         temp_new_routes);
if best_cost_improvement > $\theta_{thr}$ then
    execute best_st and update times;
    for (group, new_route) in new_routes do
        set route of group to new_route;
return
```

Table 4 Distinct types of passenger groups affected in short turn scenarios.

Type	Description	Selection
P_{-A}	current route is canceled when short turn is executed	put every passenger from canceled event by short turn in this set
P_{-B}	passengers that take advantage of delayed vehicle with no short turn	select groups changing into the delayed vehicle with slack smaller than the delay
P_{+C}	groups suffering a broken transfer when the short-turn is not executed	groups changing into delayed vehicle but without groups from (P_{-A}) and (P_{-B})
P_{+D}	groups arriving earlier at their destination with the vehicle after the short turn	any group with a final arrival event improved by short-turn, without (P_{-A}) and (P_{-B})

Algorithm 2 Evaluation Subroutine.

```
input : potential short turn st
initial_network_state ← snapshot of current network state;
init $P_{-A}$, $P_{-B}$, $P_{+C}$ as $\emptyset$;
new_routes ← $\emptyset$;
init direct_cost_improvements, costs_without_st, costs_with_st as 0;
deptime_improvement ←
    to(st).currentTime - max(to(st).regularTime, from(st).currentTime + $\theta_{dur}$);
for edge between from(st) and to(st) do
    insert groups on edge into $P_{-A}$
for event on vehicle starting with to(st) do
    for group with valid ingoing transfer at event do
        if transfer.slack ≤ min(deptime_improvement, event.currentDelay) then
            insert group into $P_{-B}$;
negative_groups ← $P_{-A} \cup P_{-B}$;
for event on vehicle starting with to(st) do
    for group ∉ negative_groups with event as final arrival event do
        // group is of type $P_{+D}$
        arrival_improvement ← min(deptime_improvement, event.currentDelay);
        find alternative route and save cost as alt_costs;
        current_costs ← group.getCurrentRoute().getRemainingCosts();
        min_no_st ← min(current_costs, alt_costs);
        min_st ← min(current_costs - arrival_improvement, alt_costs);
        direct_cost_improvements += (min_no_st - min_st);
    for group ∉ negative_groups with invalid outgoing transfer at event do
        if abs(transfer.slack) ≤ min(deptime_improvement, event.currentDelay)
         then
            insert group into $P_{+C}$;
all_affected_groups = negative_groups ∪ $P_{+C}$;
for group in $P_{+C}$ do
    reroute group;
for group ∈ $P_{-A}$ with invalid transfer do
    reroute group;
for group in all_affected_groups do
    costs_without_st += group.getCurrentRoute().getRemainingCosts();
reset network to initial_network_state;
execute short turn and update times;
for group in $P_{-A}$ do
    reroute group and save calculated route as new_route;
    insert (group, new_route) into new_routes
for group in $P_{-B}$ do
    reroute group;
for group in all_affected_groups do
    costs_with_st += group.getCurrentRoute().getRemainingCosts();
reset network to initial_network_state;
total_cost_improvement =
    direct_cost_improvements + costs_without_st - costs_with_st;
return (total_cost_improvement, new_routes)
```