Increasing Delay-Tolerance of Vehicle Schedules in Public Bus Transport

Stefan Kramkowski, Natalia Kliewer, and Christian Meier

Decision Support & Operations Research Lab, University of Paderborn, Warburger Str. 100, 33098 Paderborn, Germany, {kramkowski,kliewer,meier}@dsor.de
http://dsor.upb.de

Abstract. In public bus transport resource schedules are computed offline, but unavoidable delays occur frequently during the transportation process. Thus, schedules become infeasible and the operations control has to undertake expensive actions on the day of operations. We present different heuristic offline approaches to increase delay-tolerance of vehicle schedules. Two of these approaches are based on simulated annealing for noisy environments. The main focus is on inserting buffer times on the right place in order to cope with minor disruptions and to control delay propagation. Computational results for real-life timetables from German cities compare the approaches with regard to planned operational cost and delay-tolerance.

Key words: Robust vehicle scheduling, public bus transport, delay-tolerance.

1 Vehicle Scheduling and Disruptions

The resource scheduling for buses, the so called vehicle scheduling, is one of the main tasks in the operational planning process of public transport companies. It assigns buses to cover a given set of timetabled trips, such that planned costs are minimal. The costs consist of fixed cost per vehicle and variable cost per driven kilometer and per hour the vehicle spent outside the depot. A vehicle schedule is feasible if and only if each timetabled trip is assigned to one vehicle of the allowed vehicle-type and each vehicle starts at a depot and gets back to it at the end of the planning horizon (mostly working day). For a detailed description of the vehicle scheduling problem (VSP) and different solution approaches see [1] and [2].

Traditionally vehicle schedules are ready several weeks before the day of operations. Therefore scheduling cannot consider real driving times. Expected values for different routes and time of day are used instead for the cost-efficient resource scheduling.

Disruptions and delays are normally unavoidable on the day of operations, but the possibility of such disruptions is usually not regarded in the offline vehicle scheduling. Thus they can affect the operations of the cost-efficient schedules heavily, leading to significant increase in the costs of schedule operations.
In the last years vehicle schedules get more and more cost-efficient through application of specialized planning software and improvements of the used optimization approaches. But by decreasing the planned cost without consideration of future disruptions, the disruption-sensitivity tends to increase, because of the reduction of the idle time of buses, which can make up for delays. In occurrence of disruptions, this leads to an unscheduled assignment of additional vehicles and to penalty fees, which have to be payed to the governmental administration by the transportation company, if timetabled trip punctuality falls below an contracted minimal level (cmp. [3]). Therefore the real cost can increase contrary to the original goal, even if the planned cost has decreased.

There are different alternatives to avoid this undesirable effect: The vehicle schedules can be computed online during execution or continuously can be adjusted to the current conditions on the day of operations. This can be realized using online algorithms or by repeatedly solving recovery problems. The dynamic vehicle scheduling presented in [3] is allocated to this field as well. In the most practical cases such approaches are difficult to implement, because of the interdependencies with other planning phases, such as crew scheduling and rostering, which then also should be computed or recovered online.

To retain the established planning process, in this work another approach is persued: As before, the vehicle scheduling is accomplished offline, whereas now potential disruptions are considered in the planning process as well. By the use of this approach, an excessive reduction of vehicle idle times can be avoided. The resulting vehicle schedules are in some degree robust, or more precisely remain stable if disruptions occur. This means, the schedules are able to absorb a certain amount of delays. To the best of our knowledge, in literature exist no such approaches for bus scheduling. The terms robustness and stability, that have been introduced above for vehicle scheduling problems, are discussed in [4] in a more general context.

In section 2 four methods are presented, that implement the offline approach. Three of them are in form of a schedule-improving procedure (2.2 and 2.4) and one is schedule-constructing procedure (2.3). Section 3 compares computational results of the different approaches and gives some ideas for future research on robustness in vehicle scheduling.

2 Approaches to Increase Delay-Tolerance

After definition of a measure for delay-tolerance, we present in this section the following approaches to increase delay-tolerance:

- different flow decomposition strategies for cost-optimal flow,
- guarantee minimum buffer time before all timetabled trips,
- simulated annealing for noisy environments with two different neighbourhood operators.
2.1 Measuring Delay-Tolerance

Before describing the approaches, the term delay-tolerance has to be defined in detail. In literature (e.g. [8] and [9]) primary and secondary delays are distinguished. From the planning point of view, primary delays are exogenous and are caused directly by disruptions, whereas secondary (induced) delays are endogenous. They are evoked by primary delays through internal dependencies of the trips of one vehicle. For example, if a timetabled trip is behind schedule and the idle time before the following trip is too short and no recovery is carried out, the following trip will be secondary delayed. Only secondary delays can be influenced by modifying the vehicle schedule.

In this work secondary delays and propagated delays are used synonymously, because using recovery procedures during measuring the stability of vehicle schedules makes no sense. This means, delays are propagated until they are absorbed by idle times or the working day ends. Besides, propagated delays are only measured if they belong to timetabled trips, because delays belonging to deadheads are of least interest for the passengers and the transportation companies. Only for that reason the evaluation of stability is unaltered.

Thus, we define our measure for delay-tolerance as follows: For each timetabled trip quantify the time (in seconds) the trip is starting behind schedule. If a timetabled trip is starting punctually, this time is zero seconds. Now calculate the expected length of a secondary delay per timetabled trip \( E(SD) \) as average over all these starting time deviations.

Comparing two \( E(SD) \) values, differences can be caused by changes in frequency and/or length of the secondary delays. The actual reason can be determined via the expected length of secondary delays larger than zero \( E(SD|SD>0) \) and the probability of secondary delays larger than zero \( P(SD>0) \). They can be calculated correspondingly to \( E(SD) \). The relation between all these measures is described by equation 1.

\[
E(SD) = E(SD|SD>0)/P(SD>0) \tag{1}
\]

If secondary delays are computed not only for one delay scenario but for sundry scenarios or simulation with multiple runs is used, \( E(SD) \), \( E(SD|SD>0) \) and \( P(SD>0) \) are calculated as average over all scenarios or runs.

2.2 Decomposition Strategies

[1] recommends to model vehicle scheduling problems as time-space network, if planned cost should be minimized. In this approach the optimal solution is represented as a network flow based on arcs that has to be decomposed to a path based flow. Each path represents a workday activities for one vehicle starting and ending in a depot. Thus, various vehicle schedules can be easily computed without varying the planned cost and thereby a secondary objective can be considered. Different decomposition strategies with different objectives are presented in [5]. Furthermore, in the similar way such decomposition can be undone for every
valid vehicle schedule, in order to consider a secondary objective for an already given vehicle schedule (e.g. [6] for adaptive crew scheduling).

In this way especially schedules can be generated having identical operational cost and the same amount of total idle time, but with different distributions of the idle times after each particular bus activity. Because idle times naturally have a fundamental effect on delay-tolerance, this is the first approach analyzed in this paper.

In [5] the decomposition strategies first-in-first-out (FIFO) and last-in-first-out (LIFO) were proposed. FIFO connects the first ingoing arc of every node with the first outgoing arc, and so on. LIFO connects the last ingoing arc of every node with the first outgoing arc, and so on. Thus, the idle times are redistributed. FIFO leads to a low variance of idle times and LIFO to a high variance (cmp. figure 1). An idle time variance in between can be achieved with a random connection of ingoing and outgoing arcs (random decomposition).

![Fig. 1. Redistributing Idle Times with FIFO and LIFO Decomposition](image)

2.3 Minimum Buffer Time before Timetabled Trips

An intuitive way increasing delay-tolerance is to manually add idle times for the buses. This planned idle time is called buffer time, because it is no result of vehicle scheduling but is fixed in input of cost-efficient vehicle scheduling.

The important question is, where and how much buffer time to add: Because the timetabled trips should be protected against secondary delays, it makes sense to add buffer time before every timetabled trip. If no historical data is available, as in our case, the buffer times only can be distributed equally over all timetabled trips. Thus, a minimum idle time before every timetabled trip is guaranteed. Because the timetabled trips are scheduled to-the-minute, the buffer times should be whole minutes, too.
Simulated Annealing for Noisy Environments

Simulated annealing for noisy environments (in short SANE) is first proposed by [7]. It is a mono-criterial meta-heuristic, which can be used, if the solution objective value is subject to stochastic uncertainty. The actual objective function \( z \) is defined as:

\[
z = \text{planned cost} + \text{delay cost}
\]

\[
\text{delay cost} = \text{variable delay cost} + \text{delay penalty}
\]

\[
\text{delay penalty} = \sum \text{prop. delays} \cdot \frac{\text{fix cost per bus}}{\alpha^2}
\]

The planned cost are deterministic for each vehicle schedule, whereas the delay cost (equation 3) can be calculated only by a set of primary delays. To obtain a representative set of primary delays, Monte-Carlo simulation is used with a probability function to control the decision if a trip is delayed or not and to determine the length of possible delays. The delay penalty (equation 4) is founded by [3], who squared the length (in seconds) of each propagated delay of a timetabled trip and weights it with the fixed cost of the particular vehicle-type divided by \( \alpha^2 \). This effects, that a propagated delay of length \( \alpha \) (in seconds) is just as expensive as an additional vehicle and few small delays are preferred above one large delay. The variable delay cost represents the additional resource usage due to primary and secondary delays. As a result, the delay cost and therefore the value of the objective function are stochastically influenced, so that SANE is used as meta-heuristic scheme for this method instead of conventionally simulated annealing (cmp. [10]).

The presented method proceeds as shown in algorithm 1: Based upon a valid initial solution and an arbitrary valid neighbourhood solution, \( \sigma_{\Delta E}^2 \) is estimated. Then the initial temperature is set to the Temperature Equivalent for one sample (cmp. [7]) multiplied by \( \tau \). During the procedure, valid neighbourhood solutions are generated, evaluated and either accepted or refused, until a defined termination condition is met. In contrast to [7], in the presented approach at least 50 samples are drawn for each neighbourhood solution, because not only the difference of the fitness values but also their variance has to be estimated. The actual solution at the time of termination is taken as the result of the procedure.

The check of acceptance by Ceperley and Dewing and the sequential sampling with acceptance criterion by Glauber execute as presented in [7] with one difference: In sequential sampling a maximal sample count of 500 is introduced to speed up the procedure. According to the annealing schedule described by algorithm 1, the temperature is reduced by 5% statically every 20 iterations. The static annealing interval and the cooling factor of 0.95 have proven appropriate in several tests (see section 3). Two variants for generation of neighbourhood solutions are subsequently described in detail.
Algorithm 1: Heuristic Improvement Method for Robust Vehicle Schedules

**Input:** initial vehicle schedule $x_I$

**Result:** vehicle schedule $x_C$

current vehicle schedule $x_C \leftarrow x_I$

generate dummy neighbourhood solution $x_N \in N(x_C)$

simulate delays in $x_C$ and $x_N$ 50-times

calculate mean of objective function for $x_C$ and $x_N$

$\hat{\delta} \leftarrow E(x_N) - E(x_C)$

estimate $\sigma^2_{\Delta E}$ as variance of $\hat{\delta}$ between simulation runs

iteration count $n \leftarrow 0$

initial temperature $T_0 \leftarrow \tau \cdot \sigma_{\Delta E} \cdot \sqrt{\pi/8}$

current temperature $T_n \leftarrow T_0$

repeat

repeat

$n \leftarrow n + 1$

generate neighbourhood solution $x_N \in N(x_C)$

simulate delays in $x_N$ 50-times

calculate mean of objective function for $x_N$

$\hat{\delta} \leftarrow E(x_N) - E(x_C)$

update estimation of $\sigma^2_{\Delta E}$

if $T_n \geq \sigma_{\Delta E} \cdot \sqrt{\pi/8}$ then

check acceptance by Ceperley and Dewing criterion

else

// sequential sampling with acceptance criterion by Glauber

$m \leftarrow 50$

$P_{err}(\hat{\delta}) \leftarrow \Phi(-|\hat{\delta}| \cdot \sqrt{m/\sigma_{\Delta E}})$

while $P_{err} > P^{Glauber}_{err}(|\hat{\delta}|) \land m < 500$ do

draw another sample

$m \leftarrow m + 1$

update $\hat{\delta}$, $\sigma^2_{\Delta E}$ and $P_{err}$

end

accept better solution

end

until $n \mod 20 = 0$

$T_n \leftarrow T_{n-1} \cdot 0.95$

until $T_n < T_0/10000$
Random Based Neighbourhood Operator  Based on the current vehicle schedule, in the neighbourhood generation step, a slightly different, so called neighbourhood solution has to be computed. The random based neighbourhood operator randomly selects a timetabled trip from the current vehicle schedule and reassignes it to another randomly selected vehicle, that can serve the selected timetabled trip without time intersection or violation of vehicle-type restrictions. If no such vehicle exists, a new vehicle with the same vehicle-type as the one previously serving the selected timetabled trip is added to the schedule and the timetabled trip is assigned to the new vehicle. In any case, deadheads must be added or removed to secure validity of the modified tours. Therefore, a complete deadhead matrix is important, such that deadheads are allowed at every time of day from every stoppoint to each other. If the deadhead matrix is not complete, it has to be completed with transitiv shortest paths. Figure 2 visualizes the neighbourhood generation as a time-space network for two tours A and B, where T is the timetabled trip to be reassigned.

![Neighbourhood Generation Diagram](image)

Fig. 2. Reassigning a Timetabled Trip to another Vehicle and Adding/Removing Deadheads

Selective Neighbourhood Operator  To perform a more purposeful neighbourhood operation, another procedure has been developed. Just as the random based neighbourhood operator described above, the selective neighbourhood operator reassignes one timetabled trip to another vehicle, that is qualified to serve it. In contradiction to the random based operator, the selective operator calculates the expected propagated delay for each timetabled trip and selects that timetabled trip for reassignment with the largest propagated delay. This timetabled trip is assigend to the valid vehicle, that provides the largest buffer time straight previous to the timetabled trip. If no such vehicle exists, a new vehicle is added as described before. At last deadheads are added and removed to retain the validity of the neighbourhood schedule.

If the neighbour solution generated this way is not accepted by SANE, it is necessary to select another timetabled trip for reassignment in next neighbourhood generation. Otherwise the neighbourhood solution stays the same till it is
accepted or SANE terminates. Most appropriately, the timetabled trip with the second largest propagated delay is selected. If this new solution is declined too, the timetabled trip with the third largest propagated delay has to be choosen, and so on. If a neighbourhood solution is accepted, the next neighbourhood generation searches for the largest propagated delay again.

This procedure allows the use of an additional termination condition: If all timetabled trips are tried out for reassignment, nothing more can be done. This means, if the number of generated neighbour solutions till the last accepted is equal to the number of timetabled trips, SANE terminates.

3 Results and Outlook

The approaches were tested on three real-world problems from german cities (see table 1). For the two SANE based heuristics, vehicle schedules with zero buffer time and minimal planned cost were used as initial solutions, because we assume that the sought after robust vehicle schedules do not deviate significantly from the schedules with minimal planned cost. For the same reason \( \tau = 1 \) is used and so acceptance by Ceperly and Dewing is avoided. The simulation of primary delays during SANE uses an aggregated probability function: Based on a Bernoulli-experiment 20\% of the trips are primarily delayed and the length of the primary delays (in seconds) was sampled using a triangular distribution in the intervall \([1;600]\) with dense-maximum at 1. For the delay penalty (equation 4) \( \alpha = 1920 \) is used as in [3]. The schedules with guaranteed minimum buffer time, the initial schedules for SANE and for the decomposition strategies have been computed using the software described in [1]. The schedules with guaranteed minimum buffer time and the initial schedules for SANE are all FIFO decomposition. The initial schedules for SANE and for the decomposition strategies have zero guaranteed minimum buffer time.

Table 1. Characteristics of Test Instances

<table>
<thead>
<tr>
<th>instance</th>
<th>#timetabled-trips</th>
<th>#depots</th>
<th>#vehicle-types</th>
<th>#stoppoints</th>
<th>group-size</th>
<th>deadhead</th>
<th>density</th>
</tr>
</thead>
<tbody>
<tr>
<td>424_1_1</td>
<td>424</td>
<td>1</td>
<td>1</td>
<td>34</td>
<td>1.0</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>426_1_1</td>
<td>426</td>
<td>1</td>
<td>1</td>
<td>33</td>
<td>1.0</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>1296_1_3</td>
<td>1296</td>
<td>1</td>
<td>3</td>
<td>88</td>
<td>1.3</td>
<td>100%</td>
<td></td>
</tr>
</tbody>
</table>

The results comparing the four approaches are presented as two-dimensional diagramms (figure 3 to 5) with the planned cost as abscissa and the expected length of propagated delays per timetabled trip as ordinate (cmp. section 2.1). For each test instance one diagramm is given and the labels are all the same as

\(^1\)The instance names are encoded as: #timetabled trips.#depots.#vehicle-types. All test instances can be downloaded at http://dsor.upb.de/bustestset
in figure 3. To avoid circular reasoning, the propagated delays of the approaches solutions have been simulated using Monte-Carlo simulation with probability distribution 5 for the primary delays of each trip. This distribution is a suitable approximation of the primary delay scenarios used in [3]. To make the comparison accurate, 500 runs of the simulation have been carried out for each vehicle schedule. For detailed results see the appendix.

\[ PD \sim \lfloor \text{Exp}(\lambda) \cdot 2 \rfloor \cdot 60 \quad \text{with } \lambda = 3.3 \]  

(5)

![Diagram showing comparison of approaches for 424.1.1](image)

**Fig. 3.** Comparing Approaches for 424.1.1

As can be seen from the diagrams, delay-tolerance for all test instances can be increased by all presented approaches without disregarding the cost: The decomposition strategies improve delay-tolerance without increasing the planned cost. Thereby, FIFO decomposition (low variance of idle times) significantly performs best in most cases. Only at 1296.1.3 FIFO and random decomposition are best. A high variance of idle times (LIFO decomposition) is bad in either case.

Guaranteeing minimum buffer time before every timetabled trip decreases the expected length of secondary delays to a great extent. Introducing 5 minutes of buffer time results in almost zero secondary delays and this is the best result of all approaches. But as the delay-tolerance increases, the planned cost increases, too. This is the only approach scheduling expensive additional vehicles.

Both neighbourhood operators for SANE further improve the delay-tolerance of vehicle schedules computed with FIFO decomposition (initial solutions) but
**Fig. 4.** Comparing Approaches for 426.1.1

**Fig. 5.** Comparing Approaches for 1296.1.3
not as much as introducing one minute minimum buffer time. The increase of planned cost is very little by the use of our SANE heuristics. There are almost no differences between the results of the random based and the selective neighbourhood operator.

From a practical point of view decomposition strategies and the SANE heuristics perform best, because they decrease the expected secondary delay with no or approximately no increase of planned cost. More solutions are of interest, that close the gap towards the vehicle schedules with one minute guaranteed minimum buffer time. But this cannot be done only by changing the objective function of SANE, because some test runs suggest, that both above presented neighbourhood operators are not able to achieve vehicle schedules in this area of solution space.

Thus, we started to implement a new neighbourhood operator, that combines the power of SANE with the minimum buffer times. It adds and removes buffer times before some timetabled trips. So in every neighbourhood generation step a new vehicle schedule has to be computed. Another topic in our future research is the use of historical delay data.

References

Appendix: Detailed Testresults

Table 2 to 4 show the planned cost, the delay cost (equation 3 with $\alpha = 1920$), the probability of a secondary delay larger than zero per timetabled trip and the expected length of a secondary delay larger than zero per timetabled trip (in seconds). The expected length of a secondary delay as used in section 3 can be calculated by equation 1.

Table 2. Comparison of Decomposition Strategies

| instance | decomposition | planned cost | delay cost | P($SD > 0$) | E($SD|SD > 0$) |
|----------|---------------|--------------|------------|-------------|----------------|
| 424,1,1  | FIFO          | 2004         | 2.5%       | 75.1        |
|          | Random        | 2951679      | 3.5%       | 77.7        |
|          | LIFO          | 3764         | 4.3%       | 78.0        |
| 426,1,1  | FIFO          | 5166         | 13.8%      | 59.5        |
|          | Random        | 1934170      | 14.0%      | 60.1        |
|          | LIFO          | 6266         | 16.0%      | 60.9        |
| 1296,1,3 | FIFO          | 187142       | 5.7%       | 81.2        |
|          | Random        | 54271025     | 5.7%       | 81.0        |
|          | LIFO          | 206882       | 6.1%       | 82.6        |
### Table 3. Trade-off by Adding Minimum Buffer Times

| instance | buffer time (minutes) | planned cost | delay cost | P(SD > 0) | E(SD|SD > 0) |
|----------|-----------------------|--------------|------------|-----------|-------------|
|          | 0                     | 2951679      | 2004       | 2.5%      | 75.1        |
| 424,1_1  | 1                     | 3053943      | 418        | 0.5%      | 73.3        |
|          | 2                     | 3054066      | 139        | 0.2%      | 73.4        |
|          | 3                     | 3154915      | 37         | approx. 0%| 77.2        |
|          | 4                     | 3257680      | 20         | approx. 0%| 81.8        |
|          | 5                     | 3566095      | 21         | approx. 0%| 60.0        |
|          | 0                     | 1934170      | 5166       | 13.8%     | 59.5        |
| 426,1_1  | 1                     | 2115373      | 987        | 2.8%      | 55.1        |
|          | 2                     | 2297084      | 263        | 0.5%      | 56.4        |
|          | 3                     | 2358291      | 126        | 0.1%      | 57.0        |
|          | 4                     | 2418856      | 93         | approx. 0%| 59.3        |
|          | 5                     | 2479447      | 89         | approx. 0%| 59.2        |
|          | 0                     | 54271025     | 187142     | 5.7%      | 81.2        |
| 1296,1_3 | 1                     | 55451028     | 58839      | 1.9%      | 78.2        |
|          | 2                     | 62267636     | 15563      | 0.4%      | 76.3        |
|          | 3                     | 63476553     | 5722       | 0.1%      | 76.0        |
|          | 4                     | 66932231     | 4309       | approx. 0%| 76.6        |
|          | 5                     | 69251008     | 4112       | approx. 0%| 72.5        |

### Table 4. Comparison of Different Neighbourhood Operators for SANE

| instance | neighbourhood | planned cost | delay cost | P(SD > 0) | E(SD|SD > 0) |
|----------|---------------|--------------|------------|-----------|-------------|
| 424,1_1  | initial       | 2951679      | 2004       | 2.5%      | 75.1        |
|          | random        | 2956448      | 1874       | 2.3%      | 75.5        |
|          | selective     | 2954659      | 2036       | 2.4%      | 77.7        |
| 426,1_1  | initial       | 1934170      | 5166       | 13.8%     | 59.5        |
|          | random        | 1936114      | 4815       | 13.0%     | 59.7        |
|          | selective     | 1935075      | 5115       | 13.7%     | 59.8        |
| 1296,1_3 | initial       | 54271025     | 187142     | 5.7%      | 81.2        |
|          | random        | 54356612     | 178844     | 5.5%      | 81.2        |
|          | selective     | 54396594     | 165061     | 5.1%      | 80.5        |