Heuristic Methods for Increasing Delay-Tolerance of Vehicle Schedules in Public Bus Transport

Stefan Kramkowski* Natalia Kliewer* Christian Meier*

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

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 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 [8] and [3].

Traditionally vehicle schedules are created 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 heavily affect the operations of the cost-efficient schedules, 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 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 (see [6]). 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 problem* presented in [6] 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 [10] in a more general context.

In section 2 two methods are presented, that implement the *offline* approach. They both are in form of a schedule-improving heuristic based on *simulated annealing for noisy environments* as proposed by [2]. Section 3 compares computational results of the different approaches and two parameter-sets.

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:

Simulated annealing for noisy environments with a

- random based neighbourhood operator
- selective neighbourhood operator

2.1 Measuring Delay-Tolerance

Before describing the approaches, the term *delay-tolerance* has to be defined in detail. In literature (e.g. [4] and [11]) *primary* and *secondary* delays are distinguished. From the planning point of view, primary delays are exogenious 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 secondarily 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 ($\mathrm{E}(SD)$) as average over all these starting time deviations.

Comparing two $\mathrm{E}(SD)$ 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 ($\mathrm{E}(SD|SD>0)$) and the probability of secondary delays larger than zero ($\mathrm{P}(SD>0)$). They can be calculated correspondingly to $\mathrm{E}(SD)$. The relation between all these measures is described by equation 1.

$$\mathrm{E}(SD) = \mathrm{E}(SD|SD>0)/\mathrm{P}(SD>0) \tag{1}$$

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

2.2 Simulated Annealing for Noisy Environments

Simulated annealing for noisy environments (in short SANE) is first proposed by [2]. It is a mono-criterial meta-heuristic, which can be used, if the solutions objective value is subject to stochastic uncertainty. The actual objective function z is defined as:

$$z = \text{planned cost} + \text{delay cost} \tag{2}$$

$$\text{delay cost} = \text{variable delay cost} + \text{delay penalty} \tag{3}$$

$$\text{delay penalty} = \sum_{\text{prop. delays}} (\text{delay length})^2 \cdot \frac{\text{fix cost per bus}}{\alpha^2} \tag{4}$$

The planned cost[1] 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 [6], 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 α^2. This effects, that a propagated delay of length α (in seconds) is just as expensive as an additional vehicle and few small delays are prefered 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* (see [7]). The reason for choosing SANE instead of *stochastic annealing* (see [5]) is that it allows temperatures higher than the *Temperature Equivalent* for one sample and it controls the number of samples used for fitness evaluation (see [2]) whereby the solution process is speeded up. Sampling delays is still the most time-consuming operation in the heuristic.

[1]For a definition of the planned cost see [8].

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 $P_{err} > P_a^{Glauber}(|\hat{\delta}|) \wedge m < 500$ do
                draw another sample
                $m \leftarrow m + 1$
                update $\hat{\delta}$, $\sigma^2_{\Delta E}$ and $P_{err}$
            end
            accept better solution
        end
    until $n \bmod \beta = 0$
    $T_n \leftarrow T_{n-1} \cdot \gamma$
until $T_n < T_0/10000$
```

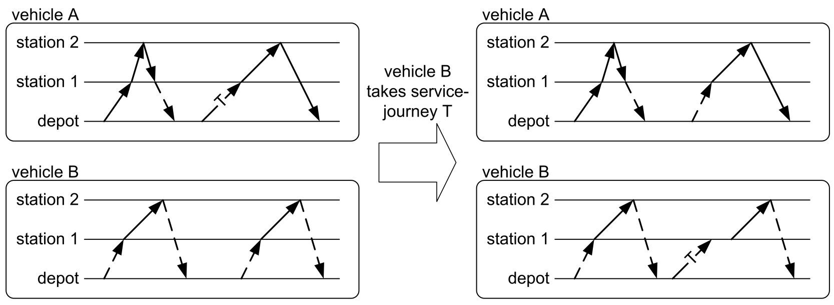

Figure 1: Reassigning a Timetabled Trip to another Vehicle and Adding/Removing Deadheads

The presented method proceeds as shown in algorithm 1: Based upon a valid initial solution and an arbitrary valid neighbourhood solution, $\sigma^2_{\Delta E}$ is estimated. Then the initial temperature is set to the *Temperature Equivalent* for one sample (see [2]) multiplied by τ. During the procedure, valid neighbourhood solutions are generated, evaluated and either accepted or refused, until a defined termination condition is met. In contrast to [2], 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 acceptence criterion by Glauber* execute as presented in [2] 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 multiplier $\gamma \in [0; 1]$ statically every β iterations. The static annealing interval and the geometric annealing have proven appropriate in several tests (see section 3). Two variants for generation of neighbourhood solutions are subsequently described in detail.

2.2.1 Random Based Neighbourhood Operator

Based on the current vehicle schedule, in the neighbourhood generation step, a slightly different neighbouring solution has to be computed. The *random based neighbourhood operator* randomly selects a timetabled trip from the current vehicle schedule and reassigns 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 all-pairs transitive shortest paths (see [1]). Figure 1 visualizes the neighbourhood generation as a time-space network (see [8]) for two tours A and B, where T is the timetabled trip to be reassigned.

2.2.2 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 reassigns 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 would stay 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).[2] As initial solutions vehicle schedules with minimal planned cost were used, because we assume that the sought after robust vehicle schedules do not deviate significantly from them. They have been computed using the software described in [8] and FIFO flow-decomposition see [9]. 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 [6].

Because Monte-Carlo simulation is used during SANE and for evaluation of the results, circular reasing has to be avoided. So final evaluation is done with probability distribution 5 for the primary delays of each trip. This distribution is a suitable approximation of the primary delay scenarios used in [6]. To make the comparison accurate, 500 simulation runs with the same pseudo random input have been carried out for each vehicle schedule.

$$PD \sim \lfloor Exp(\lambda) \cdot 2 \rfloor \cdot 60 \quad \text{with } \lambda = 3.3 \tag{5}$$

[2]The instance names are encoded as: #timetabled trips_#depots_#vehicle-types.
All test instances can be downloaded at http://dsor.upb.de/bustestset

Table 1: Characteristics of Test Instances

instance	#timetabled-trips	#depots	#vehicle-types	#stoppoints	⊘group-size	deadhead density
424_1_1	424	1	1	34	1.0	100%
426_1_1	426	1	1	33	1.0	100%
1296_1_3	1296	1	3	88	1.3	100%

Table 2: Parameter-set 1 with $\beta = 50$ and $\gamma = 0.90$

instance	neighbourhood	planned cost	delay cost	P($SD>0$)	E($SD\|SD>0$)
424_1_1	initial	2951679	2004	2.5%	75.1
	random	2956689	1693	2.2%	74.4
	selective	2953808	1956	2.4%	75.8
426_1_1	initial	1934170	5166	13.8%	59.5
	random	1936463	4829	12.7%	60.5
	selective	1935075	5115	13.7%	59.8
1296_1_3	initial	54271025	187142	5.7%	81.2
	random	54385403	176800	5.4%	80.8
	selective	54396902	166740	5.2%	80.5

The results comparing the two approaches and two different parameter-sets for β and γ are presented in table 2 and 3. For each parameter-set one table is given that compares the approaches. The tables columns from left to right are the test instances name, the used neighbourhood operator or "initial", 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.

As can be seen from the tables, delay-tolerance of all test instances can be increased by both presented neighbourhood operators and both parameter-sets without disregarding the planned cost: The increase of planned cost is very little, because no additional vehicles are used and vehicles fixed

Table 3: Parameter-set 2 with $\beta = 20$ and $\gamma = 0.95$

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

cost dominate the planned cost.

At nearly all test instances and neighbourhood operators parameter-set 1 performs better than 2 with respect to the expected length of a propagated delay (E(SD)). Regarding the planned cost, sometimes parameter-set 1 and sometimes 2 is better. Overall, many iterations at each temperature level (large β) in combination with a faster annealing (small γ) seems to be good.

Regarding delay-tolerance, the random based neighbourhood operator performs better than the selective one for the small test instances at both parameter-sets. For the larger test instance the selective neighbourhood operator more increases delay-tolerance at both parameter-sets. With respect to planned cost the ranking of neighbourhood operators is just vice versa than for delay-tolerance at all parameter-sets and test instances. In summary, the differences between the neighbourhood operators are little.

From a practical point of view the SANE based heuristics perform well, because they decrease the expected secondary delay with approximately no increase of planned cost. But more solutions are of interest, with higher delay-tolerance and higher planned cost until fixed cost of one additional vehicle. But this cannot be done only by changing the objective function of SANE, because some test runs (not shown here) suggest, that both above presented neighbourhood operatores are not able to achieve vehicle schedules in this area of solution space. We assume the solution space to be too complex for the presented simple neighbourhood operators because of the many feasibility restrictions (see 1). Thus, we started to implement some new neighbourhood operator, that use models from exact vehicle scheduling as SPP and Multi-Commodity-Minimum-Cost-Flow to cope with this.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows.* Prentice Hall, 1993.

[2] J. Branke, S. Meisel, and C. Schmidt. Simulated annealing in the presence of noise. *Journal of Heuristics*, 14:627–654, 2008.

[3] S. Bunte, N. Kliewer, and L. Suhl. An Overview on Vehicle Scheduling Models in Public Transport. *Proceedings of the 10th International Conference on Computer-Aided Scheduling of Public Transport*, Leeds, 2006.

[4] M. Carey. Ex ante Heuristic Measures for Schedule Reliability. *Transportation Research Part B*, 33:473–494, 1999.

[5] T.M.A. Fink. *Inverse protein folding, hierachical optimisation and tie knots.* University of Cambridge, Ph.D. thesis, 1998.

[6] D. Huisman, R. Freling, and A.P.M. Wagelmans. A Robust Solution Approach to the Dynamic Vehicle Scheduling Problem. *Transportation Science*, 38:447–458, 2004.

[7] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[8] N. Kliewer, T. Mellouli, and L. Suhl. A time-space network based exact optimization model for multi-depot bus scheduling. *European Journal of Operational Research*, 175:1616–1627, 2006.

[9] N. Kliewer, V. Gintner, and L. Suhl. Line Change Considerations Within a Time-Space Network Based Multi-Depot Bus Scheduling Model. In M. Hickman, P. Mirchandani, and S. Voß, editors, *Lecture Notes in Economics and Mathematical Systems: Computer-aided Systems in Public Transport*, 600:57–70, 2008.

[10] A. Scholl. *Robuste Planung und Optimierung.* Physica-Verlag, Heidelberg, 2001 (in German).

[11] P. Zhu, and E. Schneider. Determining Traffic Delayes through Simulation. In S. Voß, and J.R. Daduna, editors, *Lecture Notes in Economics and Mathematical Systems: Computer Aided Scheduling of Public Transport*, 505:387–399, 2001.