Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

SIMULATION-BASED HEADWAY OPTIMIZATION FOR A SUBWAY NETWORK: A PERFORMANCE COMPARISON OF POPULATION-BASED ALGORITHMS

David Schmaranzer

Christian Doppler Laboratory
for Efficient Intermodal Transport Operations
University of Vienna
1090 Vienna, AUSTRIA

Roland Braune
Karl F. Doerner

Department of Business Administration
Faculty of Business, Economics and Statistics
University of Vienna
1090 Vienna, AUSTRIA

ABSTRACT

We present a study on simulation-based optimization for the Viennese subway system. The underlying discrete event simulation model has several stochastic elements like time-dependent demand and turning maneuver times, direction-dependent vehicle travel and passenger travel as well as transfer times. Passenger creation is a Poisson process which uses hourly origin-destination-matrices based on mobile phone data. The number of waiting passengers on platforms and within vehicles are subject to capacity restrictions. As a microscopic element, passenger distribution along platforms and within vehicles is considered. There are trade-offs between service quality (e.g. waiting time) and costs (e.g. fleet mileage). This bi-objective optimization problem is transformed into a single-objective one by normalization and scalarization. The goal is to find optimal time-dependent headways. Computational experience is gained from 48 test instances which are based on real-world data. Several population-based evolutionary algorithms were applied. The covariance matrix adaptation evolution strategy (CMA-ES) performed best.

1 INTRODUCTION

The Viennese subway network has 5 lines and consists of 104 stations at 93 locations. There are 10 locations where 2, or – in one single case – 3 lines intersect. Table 1 contains some facts and figures on the subway system. For a schematic map of the whole subway network see Figure 2a.

During the course of a day, there are 1.23 million passenger movements. Figure 1 depicts the passenger volume over time for the currently employed solution. There are two peaks: one in the morning between 7:00 and 9:00 and a second one between 16:00 and 19:00 o'clock. The U4 has the highest passenger volume, followed by the U1, leaving the U3, followed by U6 and finally U2 as lines with lower loads. Notice that most lines have a higher morning peak between 8:00 and 9:00 than between 7:00 and 8:00 o'clock. In case of the U6, there is no significant difference, but the U1 behaves contrary (i.e., its first peak

Table 1: Facts and figures on the Viennese subway system.

line name	line color	no. of stations	line length [km]	[mi]
U1	red	19	14.54	9.04
U2	purple	20	16.86	10.48
U3	orange	21	13.40	8.33
U4	green	20	16.36	10.17
U6	brown	24	17.34	10.78
TOTAL		104	78.50	48.80

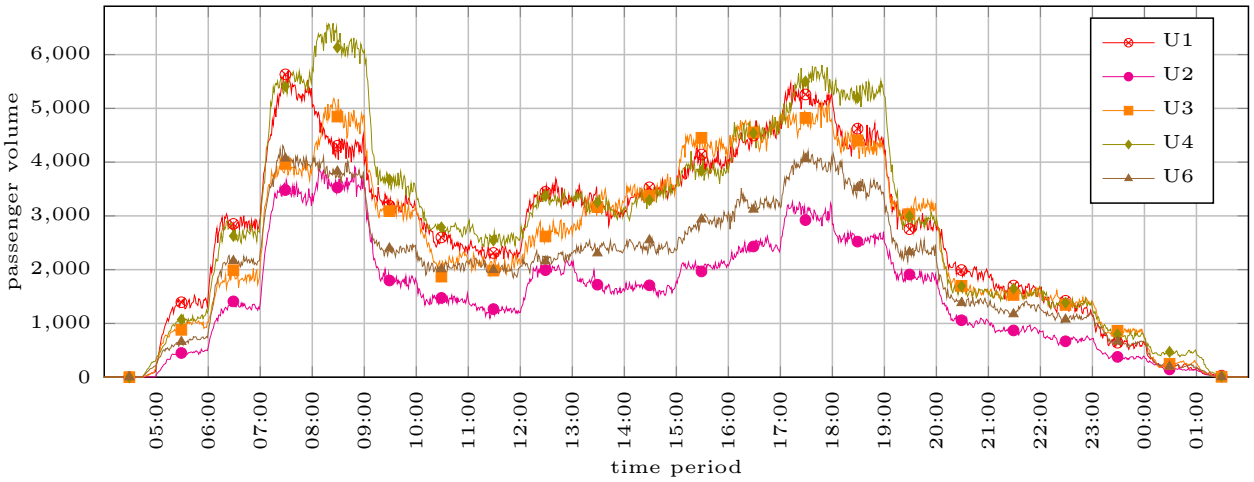


Figure 1: Passenger volume over time (simulation of currently employed headways).

behaves vice versa during aforementioned time periods). The passenger volume reaches about 22,800 in the morning, and 22,300 active passengers in the afternoon peak. About 31% perform 1 or 2 transfers.

Definition 1 The term *headway* refers to the time difference between two consecutive vehicles (e.g. 3 minutes). Its inverse, vehicles per time unit is referred to as *frequency*. The problem of *headway optimization* is also referred to as *transit network frequencies setting problem (TNFSP)*.

Vienna's population (like many cities all over the world) is growing (Hanika 2015; United Nations 2014). Urbanization and other contributing factors such as efforts to reduce CO_2 emissions, pedestrianization, improving the resident's quality of life, etc. increase demand and call for frequent re-evaluations whether provisions (i.e., tighter headways) are – now or in future – indispensable. Economic factors – namely, capital and operational expenditure (including infrastructure preservation and expansion) – are contrary to the goal of passenger satisfaction (i.e. service level). This project is dedicated to solve these conflicting goals by determining the optimal hourly headways for each line of the Viennese subway network. The subway network is only a part of the Viennese public transportation system, which includes 28 tram, 108 bus and 10 suburban railway lines. Due to the lack of data, different providers, the tram and bus lines being no competition in terms of capacity, no or limited synchronization between different types of transportation and the to expect issues when attempting to simulate not up to 5 but 151 lines, these are not considered.

We employ a simulation-based optimization approach, a concept that has already proven successful in similar application contexts (Vázquez-Abad and Zubieta 2005; Osorio and Bierlaire 2013; Osorio and Chong 2015; Chong and Osorio 2018; Ruano et al. 2017). Also, evolutionary algorithms have already been applied to this kind of problem in a similar setting (Zhao and Zeng 2006; Guihaire and Hao 2008; Yu et al. 2011). For a current review on the method, the reader is referred to Juan et al. (2015).

This paper is structured as follows: Section 2 explains the problem of headway optimization and introduces the objective function (1) as well as constraints (2). In Section 3 we describe the solution method and its building blocks, namely the discrete event simulation model and (heuristic) optimization algorithms. Next, the setup of the computational optimization experiment is introduced in Section 4. Section 5 discusses results, before Section 6 concludes the paper and presents future research.

2 PROBLEM STATEMENT

As mentioned in Section 1, there are two conflicting goals: *cost minimization* (measured in productive fleet mileage) and *service level maximization* (measured in mean waiting time per passenger). Equation 1 contains the objective function. We employ the traditional approach of normalization and weighted sum-

based scalarization, thereby transforming a bi-objective into a single-objective optimization problem.

$$\min Z = \left(\frac{m - m_{min^*}}{m_{max} - m_{min^*}} \cdot \varphi \right) + \left(\frac{w - w_{opt}}{w_{max^*} - w_{opt}} \cdot (1 - \varphi) \right). \tag{1}$$

Notation:

m	...	productive fleet mileage of the current solution
m_{min^*}	...	lowest observed productive fleet mileage (presumably near optimum)
m_{max}	...	highest productive fleet mileage at lowest possible headway
w	...	mean waiting time per passenger of the current solution
w_{opt}	...	lowest possible mean waiting time per passenger
w_{max^*}	...	highest observed mean waiting time per passenger (presumably near maximum)
φ	...	weight (i.e. ratio between fleet mileage and mean waiting time)
u_s	...	platform utilization of station s
S	...	set of stations

Note that, the first part concerning fleet mileage is deterministic (i.e., not subject to randomness). The second part of the sum concerning mean waiting time per passenger is stochastic. This is due to randomness from passenger creation (Poisson process) and other stochastic influences (e.g., vehicle travel times between stations, passenger transfer times, etc. – see Section 3.1). Therefore, the weight (φ) has an influence on the variance of the objective value Z. Since we employ not a fixed, but a varying number of replications within the simulation model (Section 3.1), this has an influence on the results.

The sole constraint type is the stations' respective platform capacity (2). In case a platform's capacity (more than 2 people per square meter) is exceeded, the solution is infeasible. The capacity of vehicles is limited as well, but this does not directly cause infeasibility: waiting passengers who are unable to board an overcrowded vehicle, continue waiting, thereby increasing the mean waiting time per passenger.

$$u_s \leq 1 \qquad \forall s \in S. \tag{2}$$

Table 2 contains the minimum and maximum values for *productive fleet mileage* and *mean waiting time per passenger* used in the objective function (1). The maximum fleet mileage (m_{max}) and optimal mean waiting time (w_{opt}) were easy to obtain: since the technically lowest possible headway is 1.5 minutes, we used this value for each line for each hour of operation and network variant (see Section 4), thereby deriving the aforementioned extreme values. Since up to 31% of passengers (depending on the network variant) perform 1 or 2 transfer operations, the optimum mean waiting time is significantly higher than the expected average waiting time per waiting process (0.75 minutes). As for lowest fleet mileage (m_{min^*}) and highest mean waiting time (w_{max^*}), we used one day long optimization runs with the covariance matrix adaptation evolution strategy, CMA-ES (see Section 3.2) and set the weight (φ) to 1 (i.e., only optimizing productive fleet mileage). This way we obtained the extreme values, which are presumably close to the optimum fleet mileage and maximum mean waiting time.

Table 2: Extreme values (m_{min^*}, m_{max}, w_{opt} and w_{max^*}), passenger data and network length per variant.

no. of	passenger volume		transfering	network length		fleet mileage [km]		waiting time [minutes]	
lines l	[million]	[%]	passengers [%]	[km]	[mi]	m_{min^*}	m_{max}	w_{opt}	w_{max^*}
5	1.23	100%	31%	78.5	48.8	13,769.67	128,738.36	1.0517	11.6664
4	1.04	85%	27%	61.6	38.3	11,612.07	101,094.52	1.0215	9.7596
3	0.68	55%	20%	48.2	30.0	8,025.90	79,115.24	0.9565	10.4336
2	0.45	37%	11%	30.9	19.2	4,188.11	50,679.28	0.8451	12.3837

3 METHODOLOGY

A subway system can be considered as a queuing network with synchronization and non-exponentially distributed service times. It is thereby too complex to merely apply analytic methods from queuing theory like Jackson networks (Jackson 1963) and its extensions. This is why we employ *simulation-based optimization* as introduced by Fu (2002). Its functional principle is as such: an optimizer (i.e., algorithm) – in our case various evolutionary algorithms – generates candidate solutions. These solutions are then handed over to a simulation model for evaluation. The result of this process (i.e., the respective solution's quality and feasibility status) is then sent back to the optimizer in order to generate new and hopefully better solutions. The next Subsections deal with the two aforementioned building blocks of simulation-optimization: the simulation model (Section. 3.1) and (heuristic) optimization algorithms (Section 3.2).

3.1 Simulation Model

The features of the discrete event simulation model are: 1) Time-dependent passenger creation (Poisson process) based on hourly origin-destination-matrices: created by the MatchMobile project (IKK 2017) using anonymous mobile phone and counting data. 2) Direction-dependent passenger transfer times (triangular distribution): several station have separate platforms (i.e., one per direction), thereby the transfer time is potentially direction-dependent. 3) Stochastic passenger times (travel time = waiting + invehicle + transfer time). 4) Direction-dependent vehicle travel times (log-normal distribution). 5) Time-dependent vehicle turning maneuver times (triangular distribution). 6) Passenger distribution along platforms and within vehicles: each platform and vehicle is divided into 3 sections (about 40 meters (44 yd) or 2 wagons per section). Provided that the station is not overcrowded, a new or transfer passenger is assigned to a platform's section and potentially boards the vehicle's corresponding section. In case the platform section in question has no free capacity, the passenger moves on to the neighboring section. In case the overcrowded section has two neighbors (i.e., the second or middle section is affected), the chance that the passenger moves to the front or end section is 50%. If a vehicle's section is overcrowded, the passenger is force to continue waiting on the platform. The distribution is based on vehicles' doors infrared counting data.

A far more detailed description, including distribution fitting and a sensitivity analysis of this discrete event simulation model can be found in Schmaranzer et al. (2016). However, it does not contain the aforementioned passenger distribution along platforms and uses older origin-destination-matrices (2012).

Since, as mentioned in Section 2, there are several stochastic elements, replications (i.e., simulation re-runs) are required to account for statistical significance. We employ a varying number of replications with a minimum of 3 and a maximum of 50 replications. The sequential evaluation process terminates once a 99.9% confidence interval with a relative error of 1% has been constructed. Since the weight (φ), used in the objective function (1), has an influence on the standard deviation of the objective value Z, the average number of replications varies, i.e., there is a negative correlation with φ.

Due to the massive number of samples (i.e., 1.23 million passenger movements), the standard deviation in mean waiting time per passenger is low. This allowed for the introduction of a "global denominator". It reduces the number of passengers as well as the capacities (platforms and vehicles) by a factor of 10. This step of course increased standard deviation, but reduced the simulation run time significantly by a factor of about 6 (0.58 instead of 3.52 seconds per run on an Intel i7-4770 with up to 3.9 GHz).

The simulation model was developed in AnyLogic 7.0.3 (64 Bit, Linux) and uses some other additional Java libraries (JGraphT 1.0.1, Apache POI 3.15 and Apache Math 3.6.1).

3.2 Heuristic Optimization Algorithms

For the purpose of optimization (i.e., finding better solutions), several population-based evolutionary algorithms are investigated. Basically, all are inspired by nature, namely theory of evolution and natural selection. A set of solutions, called population, develops over time and ideally becomes better and better. Solutions, which do not perform well, are removed form the population and replaced by new ones. This

is achieved by various evolutionary operators, for example, *crossover* (i.e., breeding new offspring based on the best solutions) and *mutation* (i.e., adding a bit of randomness to the genetic gen pool). In our case, solutions are vectors comprised of continuous values. The length of such a solution vector varies and depends on the network variant (i.e., number of lines l) and encoding type (e) – more later in the upcoming Section 4. The six employed evolutionary algorithms (i.e., optimizers) are:

- (standard) genetic algorithm (GA, see Holland 1975)
- offspring selection genetic algorithm (OS-GA, see Affenzeller et al. 2009)
- relevant alleles preserving genetic algorithm (RAP-GA, see Affenzeller et al. 2007)
- age-layered population structure genetic algorithm (ALPS-GA, see Hornby 2006)
- age-layered population structure with offspring selection genetic algorithm (ALPSOS-GA)
- covariance matrix adaptation evolution strategy (CMA-ES, see Hansen and Ostermeier 2001)

For details on these algorithms we refer to the above cited articles and books. Basically, the offspring selection genetic algorithm (OS-GA) aims for new populations where a certain percentage of offspring must be of better quality than their parents, thereby attempting to ensure progression. The RAP-GA is based on the OS-GA and allows population size alterations within certain parameters. As long as new and, with reference to the preceding population, better offspring can be created, the population size is allowed to grow up to a maximum size. The ALPS-GA employs age layers which aim at reducing premature convergence (i.e., homogeneous individuals). It continues to explore new parts of the quality landscape by creating a new sub-population of randomly generated individuals in its bottom layer. By measuring an individual's age, segregation into different age layers is performed. The goal of this scheme is to allow young individuals to develop without being dominated by older ones. When this scheme is combined with the OS-GA algorithm, it is refereed to as ALPSOS-GA. Last, the CMA-ES generates offspring not directly by crossover, but with a sophisticated sampling approach. New candidate solutions are sampled according to a multivariate normal distribution. It increases the chance of creating better offspring by constantly updating a covariance matrix which represents the pairwise dependencies between variables.

As for the 5 GA variants, the following 4 crossover operators have been used: an *average crossover* which calculates an average value out of two parents' values at the respective position of their gene material. An *arithmetic crossover* which randomly performs an average calculation or simply takes the value from the first parent. Both do not differentiate between the quality of the parents (i.e., solutions' respective quality). The *blend alpha* and *blend alpha beta crossover* (Takahashi and Kita 2001) are also used and do differentiate between better and worse parents. In both cases an interval is calculated and used as boundary for a new random value. In the case of the blend alpha crossover, the aforementioned interval can exceed only the better parents value, the other crossover can also exceed the limit set by the worse parent. For each new offspring to be created, one of these crossover operators is chosen at random.

All algorithms are aware of the currently employed headways and base the creation of the first population on it. The initial solution is the very first individual in the initial population, the remaining ones are generated by applying a normal distribution with the currently employed headways as mean and a sigma of 3.

The algorithms themselves have various parameters which need to be tuned in order to fit the problem. For this purpose, we defined a set of reasonable values for each parameter and ran a full factorial experiment, based on two hour long optimization runs on one of the smaller test instances. This tuning instance has 2 lines ($l = 2$) and an encoding of 21 parameters ($e = 21$) per line (see Section 4). Therefore, this instance has 42 optimization parameters ($l \cdot e$) in total. This network and encoding instance combination was chosen because it offered the best combination of low run time – which is due to the significantly reduced passenger volume – and still a high number of headways to be set. Table 3 contains the tuned parameter values. Notice that the resulting population size does not vary much (50 to 60). Populations sizes (up to 300) were tested, but the lower ones lead to better results – most likely due to the long evaluation time.

As for the used software: we employ several libraries from HeuristicLab 3.3.14 (Wagner et al. 2014), which is an easy-to-use metaheuristics framework developed in C#.

Table 3: Tuned parameter values for all algorithms.

parameter name	GA	OS-GA	RAP-GA	ALPS-GA	ALPSOS-GA	CMA-ES
re-evaluate elites	false	false	false	false	false	-
population size	60	50	50	50	50	60
elites in % (min. 1 elite)	20%	10%	10%	10%	10%	-
mutation probability	40%	20%	20%	20%	20%	-
selected parents	-	100	-	-	100	-
success ratio	-	0.8	-	-	0.8	-
maximum selection pressure	-	100	-	-	100	-
offspring selection before mutation	-	false	-	-	false	-
fill population with parents	-	false	-	-	false	-
min. population size	-	-	5	-	-	-
max. population size	-	-	75	-	-	-
batch size	-	-	10	-	-	-
comparison factor	-	-	0.75	-	0.5	-
effort	-	-	750	-	-	-
age gap	-	-	-	10	10	-
aging scheme	-	-	-	linear	linear	-
number of layers	-	-	-	50	50	-
age inheritance	-	-	-	0.25	0.75	-
mating pool range	-	-	-	1	1	-
plus selection	-	-	-	false	-	-
initial iterations	-	-	-	-	-	200
μ	-	-	-	-	-	10
initial σ	-	-	-	-	-	3
recombinator	-	-	-	-	-	log-weighted

4 COMPUTATIONAL EXPERIMENT SETUP

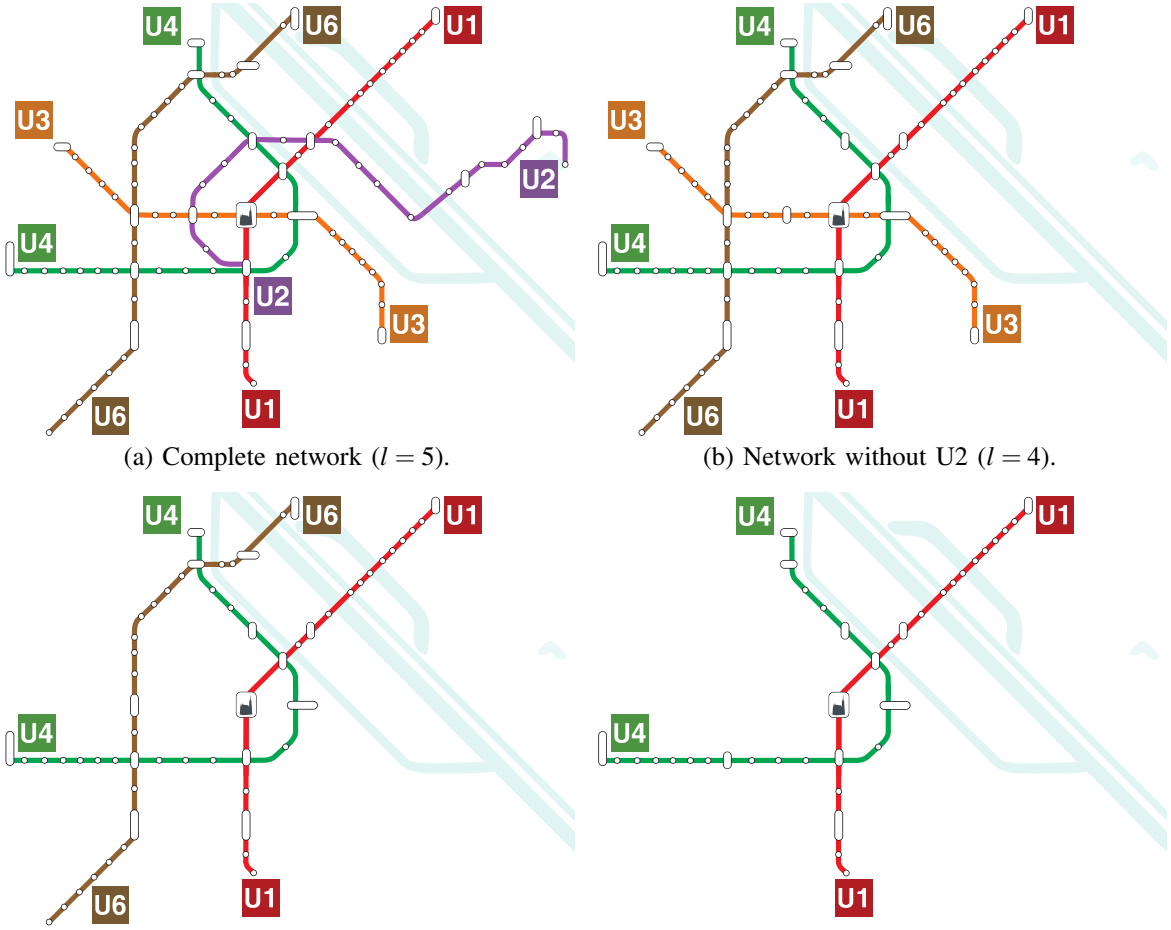
The aforementioned solution method (Section 3) is applied to 48 different test instances. These instances were created to serve several purposes: a fast instance combination for tuning the algorithms' respective parameters (Section 3.2) was required. And foremost, in order to properly compare and prove the effectiveness of a solution scheme, several different problem instances are a necessity. The instances were created by using 4 different versions of the Viennese subway network, 4 different kinds of solution encoding, and applying 3 different weights ($\varphi = 0.25, 0.50, 0.75$). The first two are described in detail in the upcoming paragraphs. Weights have already been introduced as part of the objective function (1) in Section 2.

As for *network versions*: Figure 2a contains the whole Viennese subway network. ◘ marks Stephansplatz (i.e. the city center) and its renowned landmark St. Stephen's Cathedral. Since this network version contains 5 lines, it is referred to as $l = 5$. The other variants ($l = 4$, $l = 3$ and $l = 2$; Figure 2b, 2c and 2d, respectively) are reduced versions of the full network. They were created by removing one line after another. The main criteria for this being the respective line's passenger volume (Figure 1) and area coverage. The latter being the reason why in $l = 3$ (Figure 2c) the U3 was removed instead of the U6 line.

On to *solution encoding*: Since the origin-destination-matrices change hourly, changing headways on an hourly basis too comes naturally. Given that the subway system is operating from 04:50 to 01:00 (Monday – Thursday), each line has 21 optimization parameters ($e = 21$). In order to fill and empty the lines with vehicles, their release starts at 04:30 and ends at 01:00. In the hourly encoding, the first optimization parameter of each line applies to the time period prior 05:00. Other encoding variants are 2 and 3 hour long headways, the result being 11 ($e = 11$) and 7 ($e = 7$) optimization parameters per line, respectively. The smallest version re-uses headways by means of indices and works as such: each line has merely 4 optimization parameters ($e = 4$). Those values are assigned to 21 specific time periods: {0,0,1,2,2,3,1,1,3,3,3,3,2,2,2,3,1,1,0,0,0}. The 4^{th} and 5^{th} as well as the 13^{th}, 14^{th} & 15^{th}, for example, all have an index of 2. So, the solution's line's value at this particular index is used as headway for the morning (07:00 to 09:00 o'clock) and afternoon peaks (16:00 to 19:00 o'clock).

The resulting number of optimization parameters lies between 8 and 105 ($l \cdot e$). We refer to the 8 parameter variant as *smallest* and to the 105 parameter version as *largest* instance. A run time of 6 hours proved to be sufficient for the large one. The run time of all other instances was set in relation to the number of optimization parameters (15 minutes accuracy). The smallest one has a run time of 45 minutes.

One evaluation of all 48 instances takes almost 40 hours of computation. Given, that 6 different algorithms are used, and 5 independent optimization runs (not to be confused with replications) are performed, a total of 1,200 hours is required. All experiments were conducted on the Vienna Scientific Cluster 3 (VSC 2018), which is a high performance computing (HPC) cluster comprised of 2,020 nodes, each one equipped with two Intel Xeon E5-2650v2 processors (2.6 GHz, 8 cores) and at least 64 GB RAM.



(a) Complete network ($l = 5$).

(b) Network without U2 ($l = 4$).

(c) Network without U2 & U3 ($l = 3$).

(d) Network with U1 & U4 ($l = 2$).

Figure 2: Network variants (based on the schematic plan of the Viennese Subway network as of 2014).

5 COMPUTATIONAL RESULTS

Tables 4, 5 and 6 contain the final results for all network as well as encoding variants and all 3 applied weights (φ), respectively. 5 independent and reproducible optimization runs per variant and algorithm were performed. The average objective value Z of the initial solution (i.e., the currently employed headways) served as baseline and the reported values represent average percentage deviations from that baseline. The best and second best results are highlighted **bold** and *italic* respectively. The worse are in gray.

At a weight of 0.25 (Table 4) – i.e., higher priority on mean waiting time – the CMA-ES performed best in all cases. Its average best result over all instances with this weight is -9.14%. The GA performed takes the second place in 13 out of 16 instances. RAP-GA performed worse of all in 7 out of 16.

At an equal weight of 0.50 (Table 5) again, the CMA-ES performs best. Its average best result over all instances is -6.96% and thereby not as well as with preceding weight of 0.25. The GA performs well again, but merely in 7 out of 16. Apart from the ALPSOS-GA, the other algorithms achieve 2^{nd} place more often than with a weight of 0.25. This time, the ALPSOS-GA is the worst performer in 8 out of 16 cases.

Table 4: $\varphi = 0.25$ (higher priority on mean waiting time per passenger).

no. of lines l	no. of opt. parameters	run time [hours]	diff. to initial solution's Z					
			GA	OS-GA	RAP-GA	ALPS-GA	ALPSOS-GA	CMA-ES
2	8	0.75	-4.06%	*-4.21%*	-3.99%	-4.02%	-4.10%	**-4.43%**
	14	1.50	*-4.32%*	-4.24%	-4.19%	-4.19%	-4.23%	**-4.59%**
	22	1.50	*-3.64%*	-3.40%	-3.47%	-3.47%	-3.24%	**-4.04%**
	42	3.00	*-3.78%*	-3.18%	-3.37%	-2.81%	-2.87%	**-4.54%**
3	12	1.50	*-9.17%*	*-9.17%*	-8.97%	-9.16%	-8.91%	**-9.85%**
	21	1.50	*-8.74%*	-8.23%	-8.01%	-8.53%	-8.31%	**-10.27%**
	33	2.25	*-7.87%*	-7.39%	-6.85%	-6.72%	-6.81%	**-9.40%**
	63	3.75	*-7.82%*	-6.10%	-5.70%	-4.53%	-4.65%	**-9.49%**
4	16	1.50	-10.76%	-10.65%	-10.01%	-11.44%	*-11.60%*	**-12.78%**
	28	1.50	*-9.31%*	-8.59%	-6.29%	*-9.19%*	-8.75%	**-12.58%**
	44	3.00	*-9.67%*	-9.10%	-8.07%	-8.70%	-7.51%	**-12.47%**
	84	5.25	*-9.26%*	-7.28%	-6.03%	-5.08%	-5.54%	**-12.65%**
5	20	1.50	-7.65%	-7.21%	-5.45%	-7.53%	*-7.68%*	**-9.71%**
	35	2.25	*-7.36%*	-5.63%	-5.66%	-7.17%	-6.86%	**-9.62%**
	55	3.00	*-6.52%*	-6.34%	-5.07%	-5.41%	-5.71%	**-9.96%**
	105	6.00	*-6.50%*	-4.64%	-3.55%	-2.72%	-3.84%	**-9.90%**
TOTAL			*-7.28%*	-6.58%	-5.92%	-6.29%	-6.29%	**-9.14%**

Table 5: $\varphi = 0.50$ (equal prioritization).

no. of lines l	no. of opt. parameters	run time [hours]	diff. to initial solution's Z					
			GA	OS-GA	RAP-GA	ALPS-GA	ALPSOS-GA	CMA-ES
2	8	0.75	-9.12%	*-9.24%*	-9.15%	-9.17%	-9.09%	**-9.27%**
	14	1.50	-8.33%	*-8.47%*	-8.45%	-8.42%	-8.26%	**-8.57%**
	22	1.50	-10.09%	-10.10%	*-10.17%*	-10.05%	-9.90%	**-10.49%**
	42	3.00	*-10.68%*	-10.57%	-10.65%	-10.28%	-10.29%	**-11.08%**
3	12	1.50	-5.86%	*-5.99%*	-5.84%	-5.83%	-5.91%	**-6.49%**
	21	1.50	-5.43%	-5.40%	-5.34%	*-5.53%*	-5.17%	**-5.77%**
	33	2.25	-6.56%	-6.57%	*-6.66%*	-6.53%	-6.42%	**-7.23%**
	63	3.75	*-6.77%*	-6.63%	-6.76%	*-6.80%*	-6.61%	**-7.74%**
4	16	1.50	-3.93%	*-4.09%*	-3.91%	-3.86%	-3.99%	**-4.27%**
	28	1.50	-3.98%	*-4.01%*	-3.82%	-3.88%	-3.95%	**-4.26%**
	44	3.00	-4.73%	-4.72%	-4.60%	*-4.77%*	-4.64%	**-5.49%**
	84	5.25	*-4.82%*	-4.45%	-4.56%	-4.46%	-4.58%	**-5.81%**
5	20	1.50	*-5.12%*	-4.99%	-4.97%	-5.03%	-5.00%	**-5.35%**
	35	2.25	*-5.17%*	-5.02%	-5.00%	-4.96%	-4.82%	**-5.57%**
	55	3.00	*-5.90%*	-5.77%	-5.57%	-5.50%	-5.48%	**-6.76%**
	105	6.00	*-5.72%*	-5.56%	-5.28%	-5.38%	-5.51%	**-7.22%**
TOTAL			*-6.39%*	-6.35%	-6.30%	-6.28%	-6.23%	**-6.96%**

At a weight of 0.75 (Table 6) – i.e., higher priority on productive fleet mileage – the CMA-ES performs best in all cases once more. Its average best result over all instances is -36.77%. Also the 2^{nd} place has a clear winner with the GA in all 16 instances. Like back at a weight of 0.50, the ALPSOS-GA is the worst performer in 11 out of 16. The achieved improvement is considerably higher than in the other weight variants because, Z of the initial solution correlates with φ in a positive manner. The reason for this being that the headways which are actually in effect are obviously tailored towards equality between cost (i.e., fleet mileage) and quality of service (i.e., mean waiting time), or even a bit more directed towards the latter. So at a high priority on fleet mileage ($\varphi = 0.75$) the quality of the currently employed headways is significantly lower. This makes the other weight variants (especially, $\varphi = 0.50$) more difficult to optimize.

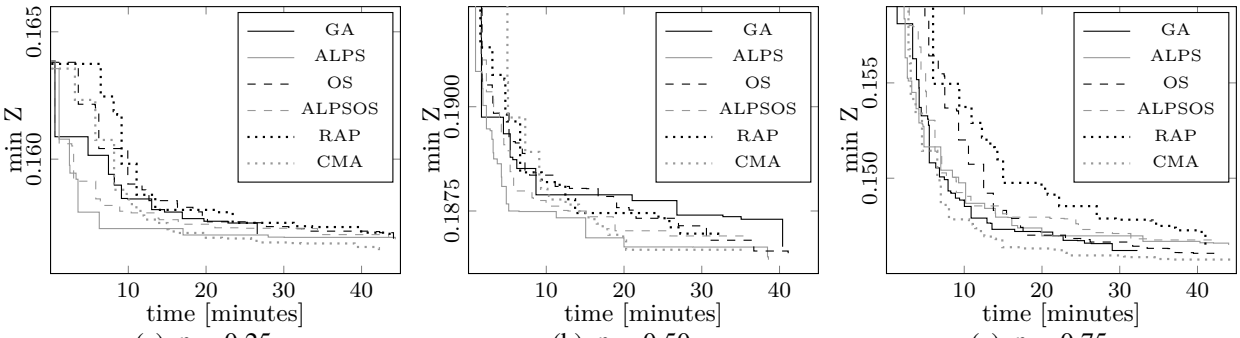
At a weight of 0.25 the CMA-ES manages to evaluate 4,740 solutions of the largest instance with 105 parameters. This number increases to 6,730 and 7,180 at a weight of 0.50 and 0.75, respectively. This effect is due to the aforementioned (Section 3.1) negative correlation between weight and number of replications. As mentioned back in Section 2, subjection of Z to randomness varies.

Table 6: $\varphi = 0.75$ (higher priority on fleet mileage).

no. of lines l	no. of opt. parameters	run time [hours]	diff. to initial solution's Z					
			GA	OS-GA	RAP-GA	ALPS-GA	ALPSOS-GA	CMA-ES
2	8	0.75	*-40.92%*	-40.86%	-40.71%	-40.66%	-40.67%	**-41.10%**
	14	1.50	*-39.74%*	-39.70%	-39.59%	-39.46%	-39.25%	**-40.00%**
	22	1.50	*-41.52%*	-41.40%	-41.29%	-40.95%	-40.57%	**-42.02%**
	42	3.00	*-41.83%*	-41.50%	-41.53%	-40.28%	-39.82%	**-42.66%**
3	12	1.50	*-35.37%*	-35.28%	-35.03%	-35.00%	-34.98%	**-36.11%**
	21	1.50	*-34.25%*	-33.76%	-33.38%	-33.13%	-33.21%	**-35.04%**
	33	2.25	*-36.05%*	-35.71%	-35.18%	-34.74%	-34.27%	**-37.12%**
	63	3.75	*-36.13%*	-34.88%	-34.46%	-33.48%	-32.98%	**-37.92%**
4	16	1.50	*-31.53%*	-31.37%	-30.79%	-31.15%	-30.47%	**-32.68%**
	28	1.50	*-29.83%*	-29.70%	-29.03%	-29.58%	-29.09%	**-31.87%**
	44	3.00	*-32.19%*	-31.55%	-31.08%	-30.90%	-30.43%	**-34.00%**
	84	5.25	*-32.42%*	-30.73%	-30.32%	-29.43%	-29.20%	**-34.86%**
5	20	1.50	*-32.81%*	-31.38%	-31.95%	-32.15%	-31.60%	**-34.80%**
	35	2.25	*-32.33%*	-30.87%	-30.65%	-30.97%	-30.06%	**-34.55%**
	55	3.00	*-33.25%*	-31.62%	-30.92%	-30.51%	-30.48%	**-36.30%**
	105	6.00	*-33.81%*	-31.50%	-30.33%	-29.45%	-29.74%	**-37.32%**
TOTAL			*-35.25%*	-34.49%	-34.14%	-33.87%	-33.55%	**-36.77%**

Figures 3 and 4 depict how the algorithms performed in the smallest ($l = 2$ and $e = 4$) and the largest ($l = 5$ and $e = 21$) test instances over time with different weights. Again, 5 independent and reproducible runs where performed. Only the algorithms' respective best run was added to the plots.

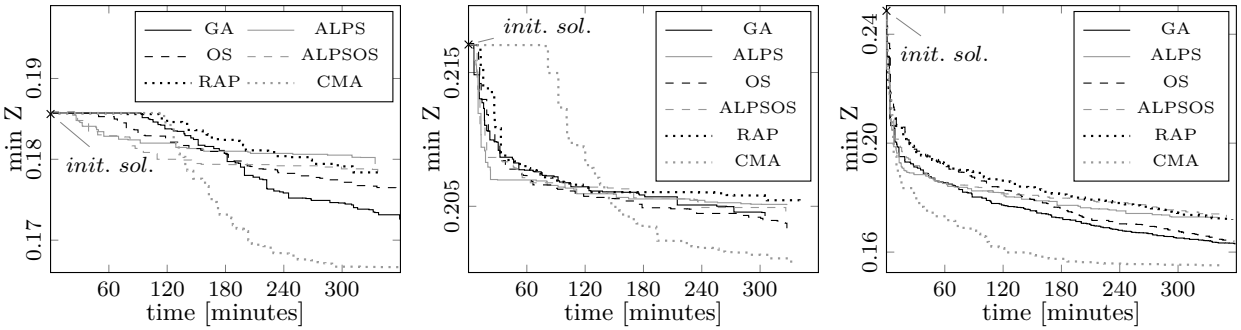
First, the smallest instance with 8 optimization parameters in total: all algorithms show a steep descent within the first ten minutes. Since – as mentioned in Section 3.2 – the initial population contains the currently employed solution, all start below the initial solution. Due to the low number of optimization parameters, this instance is easy and swiftly to optimize. Results between the algorithms (see Tables 4, 5 and 6) as well as the shape of the curves do not vary much between the 3 different weights.



(a) $\varphi = 0.25$ (b) $\varphi = 0.50$ (c) $\varphi = 0.75$

Figure 3: Best Z per algorithm over time (smallest instance with 8 optimization parameters).

The largest instance, with 105 optimization parameters, is a different situation (Figure 4): at a weight of 0.25 (Figure 4a) the algorithms (especially the GA and CMA-ES) need quite some time ($\sim$25 minutes, the GA and CMA-ES $\sim$100 minutes) to find a significantly better solution than the initial solution marked by $\times$. After about 140 minutes the CMA-ES has passed all competitors. The GA needs about 200 minutes to pass all but the CMA-ES. In Figure 4b all but the CMA-ES manage a steeper descent at the starting point and within 40 minutes. The CMA-ES needs about the same time (140 minutes) than in the previously mentioned case to catch up and exceed the other algorithms. Finally, at a weight of 0.75 (Figure 4c) all algorithms have steeper and constantly decreasing curves. This time the CMA-ES is much quicker and passes other algorithms after $\sim$20 minutes. It performed significantly better in this instance than his competitors. This can be observed in Figure 4 as well as in the previously shown tabular results (Tables 4, 5 and 6) where the results and the shapes of the curves vary stronger than in the smallest instance (Figure 3).



(a) $\varphi = 0.25$ (b) $\varphi = 0.50$ (c) $\varphi = 0.75$

Figure 4: Best Z per algorithm over time (largest instance with 105 optimization parameters).

Last, a closer investigation into the tradeoff between productive fleet mileage and mean passenger waiting time per passenger (Figure 5). The data points (i.e., coordinates) are the best results of an optimization experiment with the CMA-ES on the large instance with 105 optimization parameters where the weight (φ) was altered from 0 to 1 in steps of 0.025. The run time was increased to a whole day per optimization run. As always, 5 independent and reproducible runs per variant were performed.

The currently employed solution results in 45,909 km (28,527 mi) fleet mileage and 2.69 minutes mean waiting time per passenger. The best optimized solution with $\varphi = 0.375$ leads to 45,548 km (28,302 mi) and 2.59 minutes, thereby reducing both target measures by 0.79% and 3.72% respectively. So there is actually one with which both target values could be improved. As mentioned before, the currently employed solution is a bit closer to the equally balanced solution than the more passenger-oriented one ($\varphi = 0.25$).

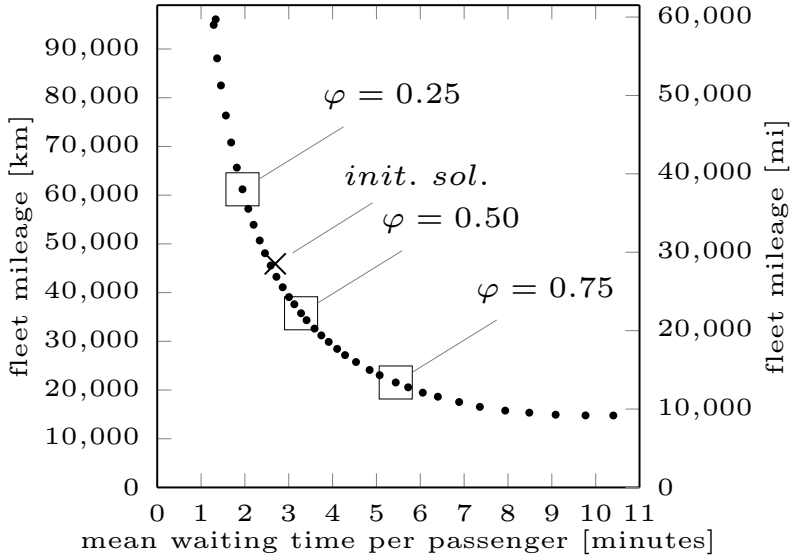


Figure 5: Fleet mileage vs. mean waiting time (large instance with 105 optimization parameters).

6 CONCLUSIONS AND PERSPECTIVES

All in all, the CMA-ES performed best in every single one of all 48 test instances. The (standard) genetic algorithm took the 2^{nd} place and produced promising solutions in all instances (2^{nd} best in 36 out of 48 cases). Both do not require many parameters and are thereby easier to tune. The ALPSOS-GA was the worse performer in 22 of 48 instances. It is likely that this is due to run time limitations and this algorithm's many parameters. Profiling runs of the algorithms' implementations within the HeuristicLab framework has shown that the "book-keeping" overhead is negligibly low. Advanced offspring selection and aging-layer concepts seem not suitable for this specific problem. Apparently, there is no danger in

terms of premature convergence and the aforementioned solution schemes do not manage to produce the necessary number of generations within the run time limit. However, in small instances the difference between these populations-based algorithms seems almost negligible. In larger instances, especially with unequal weights, the gap increases. The equally weighted instances were more difficult to improve due to the relatively good performance of the initial solution. The currently employed headways are close to equality but a bit more tailored towards quality of service (i.e., mean waiting time). The optimized solution at a weight of 0.375 offers both, cost reduction of 0.79% and service quality improvement of 3.72%.

As for the future, we are going to develop problem-specific crossover operators in order to improve the results. Furthermore, we intend to test the use factors instead of values – possibly a combination of both – as optimization parameters and use discrete instead of continuous encoding. Also, different types of metaheuristics or a Pareto-based solution approach could be an option to solve the problem. Other extensions concern the implementation of planned disruptions. For the time being, the real world system is still unaffected by this study. Possible future impacts are changes in the lines' respective hourly headways (i.e. new schedule), planning of vehicle acquisition, infrastructure alterations, disruption management, etc.

ACKNOWLEDGMENTS

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development as well as Wiener Linien GmbH & Co KG (i.e., the Viennese urban public transportation provider) is gratefully acknowledged. The authors would also like to thank the reviewers for their careful reading of this paper and for their helpful comments and suggestions.

REFERENCES

Affenzeller, M., S. Wagner, and S. Winkler. 2007. "Self-adaptive Population Size Adjustment for Genetic Algorithms". In *Computer Aided Systems Theory EUROCAST 2007*, edited by R. M. D. et al., Volume 4739, 820–828. Berlin, Heidelberg: Springer.

Affenzeller, M., S. Winkler, S. Wagner, and A. Beham. 2009. *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Number 6 in Numerical insights. Boca Raton: Chapman & Hall/CRC Press.

Chong, L., and C. Osorio. 2018. "A Simulation-Based Optimization Algorithm for Dynamic Large-Scale Urban Transportation Problems". *Transportation Science* 52(3):637–656.

Fu, M. C. 2002. "Optimization for Simulation: Theory vs. Practice". *INFORMS Journal on Computing* 14(3):192–215.

Guihaire, V., and J.-K. Hao. 2008. "Transit Network Design and Scheduling: A Global Review". *Transportation Research Part A: Policy and Practice* 42(10):1251–1273.

Hanika, A. 2015. "Zukünftige Bevölkerungsentwicklung Österreichs und der Bundesländer 2014 bis 2060 (2075)". *Statistische Nachrichten* (1):12–33.

Hansen, N., and A. Ostermeier. 2001. "Completely Derandomized Self-Adaptation in Evolution Strategies". *Evolutionary Computation* 9(2):159–195.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. 1 ed. Complex Adaptive Systems. Cambridge, MA: MIT Press.

Hornby, G. S. 2006. "ALPS: The Age-layered Population Structure for Reducing the Problem of Premature Convergence". In *GECCO '06 Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 815–822: ACM Press.

IKK 2017. "Matchmobile – Multimodal Trip Chains from Mobile Phones". Accessed Jun. 20, 2017. http://www.ikk.at/en/projekt/matchmobile-multimodal-trip-chains-from-mobile-phones.

Jackson, J. R. 1963. "Jobshop-like Queueing Systems". *Management Science* 10(1):131–142.

Juan, A. A., J. Faulin, S. E. Grasman, M. Rabe, and G. Figueira. 2015. "A Review of Simheuristics: Extending Metaheuristics to deal with Stochastic Combinatorial Optimization Problems". *Operations Research Perspectives* 2:62–72.

Osorio, C., and M. Bierlaire. 2013. "A Simulation-based Optimization Framework for Urban Transportation Problems". *Operations Research* 61(6):1333–1345.

Osorio, C., and L. Chong. 2015. "A Computationally Efficient Simulation-based Optimization Algorithm for Large-scale Urban Transportation Problems". *Transportation Science* 49(3):623–636.

Ruano, E., C. Cobos, and J. Torres-Jimenez. 2017. "Transit Network Frequencies-Setting Problem Solved Using a New Multi-Objective Global-Best Harmony Search Algorithm and Discrete Event Simulation". In *Advances in Soft Computing*, edited by O. Pichardo-Lagunas and S. Miranda-Jiménez, Volume 10062, 341–352. Cham: Springer International Publishing.

Schmaranzer, D., R. Braune, and K. F. Doerner. 2016. "A Discrete Event Simulation Model of the Viennese Subway System for Decision Support and Strategic Planning". In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. K. Roeder et al., 2406–2417. Piscataway, New Jersey: IEEE.

Takahashi, M., and H. Kita. 2001. "A Crossover Operator using Independent Component Analysis for Real-coded Genetic Algorithms". In *Proceedings of the 2001 Congress on Evolutionary Computation*, Volume 1, 643–649: Institute of Electrical and Electronics Engineers, Inc.

United Nations 2014. *World Urbanization Prospects: The 2014 Revision (Highlights)*. United Nations.

Vázquez-Abad, F. J., and L. Zubieta. 2005. "Ghost Simulation Model for the Optimization of an Urban Subway System". *Discrete Event Dynamic Systems* 15(3):207–235.

VSC 2018. "Vienna Scientific Cluster". Accessed Apr. 03, 2018. http://vsc.ac.at/systems/vsc-3.

Wagner, S., G. Kronberger, A. Beham, M. Kommenda, A. Scheibenpflug, E. Pitzer, S. Vonolfen, M. Kofler, S. Winkler, V. Dorfer, and M. Affenzeller. 2014. "Architecture and Design of the HeuristicLab Optimization Environment". In *Advanced Methods and Applications in Computational Intelligence*, edited by R. K. et al., Volume 6 of *Topics in Intelligent Engineering and Informatics*, 197–261. Heidelberg: Springer International Publishing.

Yu, B., Z. Yang, X. Sun, B. Yao, Q. Zeng, and E. Jeppesen. 2011. "Parallel Genetic Algorithm in Bus Route Headway Optimization". *Applied Soft Computing* 11(8):5081–5091.

Zhao, F., and X. Zeng. 2006. "Optimization of Transit Network Layout and Headway with a Combined Genetic Algorithm and Simulated Annealing Method". *Engineering Optimization* 38(6):701–722.

AUTHOR BIOGRAPHIES

DAVID SCHMARANZER is a predoctoral researcher and PhD student at the *Christian Doppler Laboratory for Efficient Intermodal Transport Operations* (University of Vienna). He holds master degrees in operations and in supply chain management from Upper Austria University of Applied Sciences Steyr. His research interests are simulation and combinatorial optimization. His email address is david.schmaranzer@univie.ac.at.

ROLAND BRAUNE is a postdoctoral researcher at the *department of business administration* (University of Vienna). He holds a PhD in computer science from Johannes Kepler University Linz. His research interests include combinatorial optimization problems, scheduling problems, branch and bound, problem-specific heuristics, constraint programming and metaheuristics. His email address is roland.braune@univie.ac.at.

KARL F. DOERNER is a full professor at the *department of business administration* (University of Vienna). He is head of the *Christian Doppler Laboratory for Efficient Intermodal Transport Operations* and core member of research platform *data science @ University of Vienna*. He holds a PhD in business informatics and the venia legendi in business administration. His main research interests focus on the development of intelligent search and optimization techniques based on meta- and matheuristics primarily for decision problems in logistics and transportation. His email address is karl.doerner@univie.ac.at.