Multi-objective simulation optimization for complex urban mass rapid transit systems

David Schmaranzer^{1,2} · Roland Braune¹ · Karl F. Doerner^{1,2,3}

Published online: 25 September 2019
© The Author(s) 2019

Abstract

In this paper, we present a multi-objective simulation-based headway optimization for complex urban mass rapid transit systems. Real-world applications often confront conflicting goals of cost versus service level. We propose a two-phase algorithm that combines the single-objective covariance matrix adaptation evolution strategy with a problem-specific multi-directional local search. With a computational study, we compare our proposed method against both a multi-objective covariance matrix adaptation evolution strategy and a non-dominated sorting genetic algorithm. The integrated discrete event simulation model has several stochastic elements. Fluctuating demand (i.e., creation of passengers) is driven by hourly origin-destination-matrices based on mobile phone and infrared count data. We also consider the passenger distribution along waiting platforms and within vehicles. Our two-phase optimization scheme outperforms the comparative approaches, in terms of both spread and the accuracy of the resulting Pareto front approximation.

Keywords Multi-objective optimization · Simulation optimization · Headway optimization · Transit network frequencies setting problem · Population-based algorithms · Multi-directional local search · Public transportation

1 Introduction

In 1950, 29.6% of the world's population lived in urban areas. Since then, this percentage has increased every year, reaching 55.3% in 2018. The United Nations (2018) expects this

✉ David Schmaranzer
david.schmaranzer@univie.ac.at

Roland Braune
roland.braune@univie.ac.at

Karl F. Doerner
karl.doerner@univie.ac.at

¹ Department of Business Decisions and Analytics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

² Christian Doppler Laboratory for Efficient Intermodal Transport Operations, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

³ Data Science @ Uni Vienna, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

trend to continue, such that by 2050, an estimated 68% of the world's population will be living in urban areas. North America and Europe already have reached 82.2% and 74.5% urbanization, respectively; by 2050, those values will likely be 89% in North America and 83% in Europe. In line with these trends, Vienna's population—like that of many cities all over the world—is growing and likely will exceed two million by 2025 (Statistik Austria 2017; Hanika 2018). Such population growth, combined with traffic congestion, efforts to reduce emissions, and municipal ambitions to improve the quality of life for residents (e.g., pedestrianization, reducing auto mobile as well as truck traffic) and tourists, make it necessary to readjust complex urban public transportation systems constantly. Strategic and tactical planners must determine whether existing provisions are effective, now and in the future.

Complex urban mass rapid transit systems (e.g., subway, metro, tube, underground, heavy rail) are a critical component of successful cities. They were introduced to improve movement in urban areas and reduce congestion. These motives remain valid but—as initially mentioned—also are reinforced by pedestrianization, tourism, and environmental demands. Such networks are ubiquitous, such that 25% of cities with a population of one million, 50% with two million people, and all cities with a population above ten million host them (Roth et al. 2012). In some cities, they have existed for more than a century, and there are significant similarities among the different networks, despite the unique cultures, economies, and historical developments in each city. That is, most networks consist of a set of stations delimited by a *ring-shaped core* from which *branches* extend beyond the city. Subway networks also tend to experience similar peak times, in the morning and afternoon, reflecting increased demand by employees traveling to or from their workplaces (Sun et al. 2014). In turn, the same issue confronts all subway systems: fluctuating demand must be satisfied by adjusting and re-adjusting transport capacity.

The term *headway* refers to the time difference between two consecutive vehicles. Changes to headway (or its inverse, *frequency*, defined by vehicles per unit of time) might lead to overcrowded train stations and vehicles, unless they are carefully planned. This challenge constitutes the *transit network frequencies setting problem (TNFSP)*, for which the solution demands a balance between capital and operational expenditures (e.g., infrastructure preservation, potential expansion) with passenger satisfaction (i.e., service level). A balance between these conflicting goals produces optimal time-dependent headways for each passenger line.

The planning process for public transportation usually proceeds in the following order: (1) network and line planning, (2) frequency (i.e., headway) setting, (3) timetabling, (4) vehicle scheduling, (5) duty scheduling, and (6) crew rostering (Ceder and Wilson 1986; Ceder 2001; Guihaire and Hao 2008; Liebchen 2008). In typical planning approaches, the earlier planning stage provides the input for the subsequent tasks. The second step provides the optimization of headways. Comprehensive surveys of public transport planning are available in Guihaire and Hao (2008), Farahani et al. (2013), and Ibarra-Rojas et al. (2015). For example, Guihaire and Hao (2008) and Ibarra-Rojas et al. (2015) review 26 contributions dealing with headway optimization, showing that early works assume fixed demand and are based on analytic models (Newell 1971; Salzborn 1972; Schéele 1980; Han and Wilson 1982). Six of the 26 works (Shrivastava et al. 2002; Shrivastava and Dhingra 2002; Yu et al. 2011; Huang et al. 2013; Li et al. 2013; Wu et al. 2015), as well as Ruano et al. (2017) more recently, employ genetic algorithms for the headway optimization problem. Most of them also use non-linear models, though Li et al. (2013) offer a simulation model. Evolutionary algorithms have been applied in similar settings (Zhao and Zeng 2006; Guihaire and Hao 2008; Yu et al. 2011), though only a few contributions address time-dependent demand (Niu and Zhou 2013; Sun et al. 2014). Herbon and Hadas (2015) focus on morning and afternoon peaks and use a

generalized newsvendor model. Unlike studies that attempt to optimize a single line (e.g., the latter two and Ceder, 1984, 2001), we seek to optimize several lines (i.e., a whole network) at once (Yu et al. 2011). Following sparse contributions (Vázquez-Abad and Zubieta 2005; Mohaymany and Amiripour 2009; Ruano et al. 2017) that apply simulation to a problem-specific context, we turn to simulation optimization. This approach has proven successful in similar application contexts (Osorio and Bierlaire 2013; Osorio and Chong 2015; Chong and Osorio 2018).

Several problems arise when devising a model of a complex service system like an urban mass rapid transit network. The data pertaining to the structure of an existing transportation system (e.g., subway network's lines and stations) are relatively easy to obtain, but passenger data are not. To model demand, we need to know how many passengers want to travel from one specific location to another and when. Prior contributions use count data (Ceder 1984), smart card data (Pelletier et al. 2011), or mobile phone data (Friedrich et al. 2010), and various technologies can support pedestrian counting and tracking (Bauer et al. 2009). We employ hourly origin-destination-matrices, originally created by the MatchMobile project (IKK 2017), and infrared count data. It also is difficult to gauge passenger behavior and how they decide which route to take (Agard et al. 2007). Raveau et al. (2014) identify regional differences in passenger behavior too. For reviews on route choice, the reader is referred to Bovy and Stern (1990) and Frejinger (2008).

This article seeks to add to this line of literature, by making several contributions. We propose a multi-objective simulation-based headway optimization scheme for complex urban mass rapid transit systems that is inspired by a real-world case but generic enough to fit other cities' complex rail-bound public transport systems. Three evolutionary algorithms, applied to solve the associated headway optimization problem, are embedding in a simulation optimization framework. In addition to two traditional multi-objective evolutionary algorithms, we apply a newly developed two-phase algorithm. The first phase is a single-objective covariance matrix adaptation evolutionary strategy (SO-CMA-ES), and the second is a problem-specific multi-directional local search (MD-LS). Finally, we conduct a comprehensive computational evaluation, using test instances based on a real-world subway network.

Section 2 contains an introduction to the Viennese subway system and presents the objective functions (including constraints). In Sect. 3, we describe the solution method and its building blocks, namely, the discrete event simulation model and (heuristic) optimization algorithms. Next, Sect. 4 explains the computational experiment setup and the process of tuning the applied algorithms' respective parameters, then Sect. 5 presents the results. Last, Sect. 6 concludes and proposes some research directions.

2 Problem statement

We introduce the Viennese subway system (including network features and passenger volume) in Sect. 2.1. Section 2.2 details the objective functions, alternatives, and the decision process.

2.1 The Viennese subway network

The Viennese subway network is a complex mass rapid transit system. As of 2016, it had a total length of 78.5 km and spanned five lines. Figure 1 depicts the whole subway network, with crossing stations marked with abbreviated station identifiers. There are 104 stations

Fig. 1 Schematic plan of the Viennese subway network as of 2016 (crossing stations are marked with abbreviated station identifiers)

Table 1 Facts and figures on the Viennese subway system

Line name	Line color	No. of stations	Line length (km)	Ø station distance (m)
U1	Red	19	14.54	808
U2	Purple	20	16.86	887
U3	Orange	21	13.40	670
U4	Green	20	16.36	861
U6	Brown	24	17.34	754
Total		104	78.50	793

at 93 locations; at ten of them, two or three (single-case, Karlsplatz KP) lines intersect. The abbreviation SC marks Stephansplatz (i.e., the city center) and its renowned landmark St. Stephen's Cathedral. The U5 line will be constructed between 2019 and 2024. Table 1 contains some facts and figures about the Viennese subway system.

During the course of a regular weekday (work and school day from Monday to Thursday), there are 1.37 million passenger movements. Figure 2 depicts a simulation of the passenger volume with the currently employed solution. There are two peaks: between 7:00 and 9:00 and between 16:00 and 19:00. The U1 and U3 lines carry the highest passenger volume, followed by U4 and U6; U2 has the lowest passenger load. The U1 has a higher morning peak between 7:00 and 8:00, whereas the others' highest peak is between 8:00 and 9:00. Then during the afternoon peak, U3 experiences the highest volume. This is due to passengers not only moving from home to work and then straight back again, but head to other destinations here and there. Between the U3 stations Westbahnhof (W) and Volkstheater (VT), a pedestrianized shopping area called Mariahilfer Straße serves as an alternate stop for many passengers. The passenger volume reaches 24,000 in the 2-h morning and 23,000 in the 3-h afternoon peak.

Fig. 2 Passenger volume over time (simulation of currently employed headways)

2.2 Objective functions and constraints

As mentioned in Sect. 1, there are two conflicting goals: *cost minimization* (measured in productive fleet mileage m) and *service level maximization* (measured in average waiting time per passenger, denoted w). We seek an appropriate trade-off that achieves the optimal hourly headway for each line of the Viennese urban mass rapid transit system. The productive fleet mileage m is defined as the sum of all vehicle runs (i.e., how often vehicles were released) times their respective lines length, so it does not contain unproductive mileage such as turning maneuver distances. Equations 1 and 2 contain the objective functions for these aforementioned target measures; because of their different scale (tens of thousands of kilometers of fleet mileage versus a few minutes of average waiting time), we employ typical normalization and scalarization approaches. We must rely on a stochastic simulation model for the solution evaluation, so we can only approach Pareto-optimal solutions, without guaranteeing exact optimality. Therefore, our goal is to determine a good *approximation set* of the Pareto front, that is, a set of objective vectors (each corresponding to a solution candidate), such that no vector in that set dominates another one in the same set (Zitzler et al. 2003). An objective vector *dominates* another vector if it is at least as good as the latter in all objectives, and better for at least one objective.

$$\min Z_1 = \frac{m - m_{\min^*}}{m_{\max} - m_{\min^*}}. \quad (1)$$

$$\min Z_2 = \frac{w - w_{\text{opt}}}{w_{\max^*} - w_{\text{opt}}}. \quad (2)$$

Because the first phase of our two-phase algorithm (Sect. 3.2) is a single-objective algorithm, we apply the traditional weighted sum-based approach and transform a bi-objective into a single-objective optimization problem (Eq. 3). This approach has been used previously as well (Schmaranzer et al. 2018, 2019); for this study its use is limited to the first phase of our optimization scheme and the determination of the required minimum/maximum objective values (Sect. 4). Table 2 lists the notation.

$$\min Z = Z_1 \cdot \varphi + Z_2 \cdot (1 - \varphi). \quad (3)$$

Table 2 Notation

m	Productive fleet mileage of the current solution
m_{min^*}	Lowest observed productive fleet mileage
m_{max}	Highest productive fleet mileage at lowest possible headway
w	Average waiting time per passenger of the current solution
w_{opt}	Lowest average waiting time per passenger
w_{max^*}	Highest observed average waiting time per passenger
φ	Weight (i.e., ratio between fleet mileage and average waiting time)
S	Set of stations
P_s	Set of platforms at station s
u_{sp}	Utilization of platform p at station s
l	Number of lines (i.e., network variant; see Sect. 4.1)
d	Number of decision variables per line (see Sect. 4.1)

Equation 1, related to fleet mileage, is deterministic (i.e., not subject to randomness). Equation 2, regarding the average waiting time per passenger, is stochastic, due to the randomness of passenger creation (Poisson process) and other stochastic influences (e.g., vehicle travel times between stations, passenger transfer times; see Sect. 3.1). Replications (i.e., simulation re-runs of the model; Sect. 3.1) are required to account for statistical significance. We employ a varying number of replications (see Sect. 4 and Algorithm 4), based on the weight given to the waiting time component of the objective. In the single-objective case (i.e., using Eq. 3), its average varies, and there is a negative correlation with φ . Therefore, weight φ influences the variance of the objective value Z . In a multi-objective setting—featuring the direct use of Eqs. 1 and 2, where only the latter is of consequence—the average number of replications does not vary. The sole constraint type is the stations' respective platform utilization u_{sp} . If a platform's capacity (two people per square meter) is exceeded, the solution is infeasible. The capacity of vehicles is limited too, but it does not directly cause infeasibility, because waiting passengers who are unable to board an overcrowded vehicle continue waiting, increasing the average waiting time w per passenger, which reduces the solution's quality in that regard.

Guihaire and Hao (2008) and Ibarra-Rojas et al. (2015) cite several other objectives and constraints applied in similar works. A typical determinant of headway optimization is fleet size. Its minimization can be used as an objective (Salzborn 1972), and its current or maximum could be used as a constraint (Furth and Wilson 1981; Han and Wilson 1982). Because the demand is fluctuating, we decided, in collaboration with the Viennese public transportation provider, against using the fleet size as an objective. Similarly, vehicle runs measure the number of times a line's vehicles go back and forth (Ceder 1984). Unlike fleet size, it is not driven by peak hours. Unfortunately it also disregards the lines' respective lengths. Because of these reasons we decided to use the productive fleet mileage as an objective from the transportation provider's view (Eq. 1). Because the transportation provider uses proportional cost rates (i.e., operating and maintenance cost per productive kilometer), the overall total cost can be immediately derived from the mileage.

In terms of the customer's view (i.e., service level), some studies use passenger-related total or average times like travel time (Schéele 1980; Dollevoet et al. 2015) or waiting time (Furth and Wilson 1981). An extensive sensitivity analysis of a previous version of the simulation model (Schmaranzer et al. 2016) revealed that the travel time is only driven by the waiting time, and a passenger's invehicle time increases only in bunching situations (i.e., when there

are too many active vehicles in a line, so they start blocking and slowing down one another). The current model ensures a minimum headway (i.e., minimum on the time before another vehicle can be released), so this situation is no longer likely, and instead the invehicle time remains consistent, as does the transfer time. That is, alighting passengers do not use space on the waiting platform, and there are no limitations on concurrent transfer operations. For more details on both, see Sect. 3.1.

3 Methodology

A subway system constitutes a queuing network with synchronization and non-exponentially distributed service times. It is too complex to be investigated using simple analytic methods from queuing theory like Jackson networks (Jackson 1963) and its extensions. Instead we apply *simulation optimization*, as introduced by Fu (2002). For current reviews on the general method, the reader is referred to Juan et al. (2015) and Amaran et al. (2016). As a functional principle (Fig. 3), an optimizer or algorithm—in our case an evolutionary algorithm—generates candidate solutions that are handed over to a simulation model for evaluation. The result of this process (i.e., solution quality and feasibility status) then reenters the optimizer, which generates new and hopefully better solutions. Section 3.1 describes the core of the optimization approach, or the simulation model itself. The optimization-related part, in particular our proposed bi-objective approach, is the subject of Sect. 3.2.

Fig. 3 Functional principle of simulation optimization (Fu 2002)

3.1 Discrete event simulation model

This section contains a brief description of the simulation model of the Viennese network as of 2016. Detailed model descriptions can be found in Schmaranzer et al. (2016, 2018, 2019). The second paper contains an earlier version using data from 2014. The first published article uses not only older data (from 2012) and does not consider passenger distribution along platforms and within vehicles.

The basic functional principle of a complex urban mass rapid transit system can be modeled by a directed graph, as in Fig. 4 for two lines. The three interacting entities are the *stations*, *vehicles*, and *passengers* in each line. Both lines contain five stations each (s_0 to s_4 and s_5 to s_9). Vehicles only move on the black continuous arcs. At the end-of-line stations (s_0 , s_4 , s_5 , and s_9), additional arcs allow vehicles to perform a turning maneuver and possibly start anew. With the exception of stations s_2 and s_7 , all station have their own geographical location g ; the others intersect at g_2 . In this case, additional directed arcs (italics, dotted) allow passengers to move to another line's waiting platform. Initially, each passenger has a geographic origin and a destination (e.g., g_0 to g_8) that defines a path that contains the crucial stations (e.g., s_0 , s_2 , s_7 , s_9). The path allows passengers to know where to begin their journey

Fig. 4 Example of two intersecting lines (directed graph)

(s_0) and where to exit a vehicle, because they have reached their final destination (s_9) or must perform a transfer (s_2 to s_7).

Each *station* is assigned to a specific line and has either one island platform or two separate platforms. The island platform variant potentially serves and shares its capacity with both directions. If there are separate platforms, or an island platform has an impassable wall in the middle, half of its capacity applies to each direction. The model distinguishes between platforms with and without shared capacity. Among 104 stations, 47 have an island platform without a physical barrier, and the remaining 57 have either two separate platforms or an island platform with a physical barrier, creating separate capacities. The platform capacity is two people per square meter, and it depends on its surface area (excluding safety distance between the waiting passengers and moving vehicles). As mentioned in Sect. 2.1, ten of 93 locations have two or three (single-case) stations, allowing passenger transfer between lines. Waiting platforms—as well as the upcoming vehicle entities—are about 120 m long. To account for passenger distribution, both are divided into three sections (front, middle, back) of about 40 m, or two wagons each. We included this factor at the request of the Viennese public transportation provider.

The creation of *passenger* entities is driven by a time-dependent Poisson process (i.e., *exponential* distribution) that starts at 04:45 and continues until 01:00. It requires hourly origin-destination-matrices; we use matrices based on those originally created by the Match-Mobile project (IKK 2017) and infrared count data. The former are based on anonymous mobile phone data from the year 2014. Our industrial partner provided count data from 2016, which we used to create new, updated versions of the origin-destination-matrices. For the precise procedure, see Schmaranzer et al. (2019). Origin-destination-pairs represent the number of passengers who wish to travel, within a certain time frame (e.g., from 08:00 to 08:59), from one geographical location to another. As described in the example at the beginning of this section (Fig. 4), each passenger is assigned a path, generated by Dijkstra's shortest path algorithm (Dijkstra 1959). Every line of the Viennese mass rapid transit system (Fig. 1) can be reached from another line by up to two transfer operations, so the penalty weights for these arcs (see *passenger transfer* in Fig. 4) were set to 1400 m and 3.9 min. Without these penalty weights, passenger would be tempted to perform unnecessary transfers (i.e., more than two). The weight for non-transfer arcs is the symmetric distance between stations or the average of the two mean values of the vehicle travel times between two stations. Full factorial experiments revealed that when 85% of passengers make their path decisions based on distance, and 15% base it on time, we obtain the smallest overall difference from the infrared count data; this combination is also deemed realistic by our industrial partner. Simulation on the full Viennese network (Fig. 6b) using the currently employed headways reveals that about 37% of all passengers perform one or two transfer operations. A passenger's journey

begins at a station's waiting platform and ends at the final destination. Provided that the platform still has free capacity, a new or transfer passenger gets assigned to a platform's section and potentially boards the vehicle's corresponding section. The passenger distribution along platforms is derived from the vehicles' doors' infrared count data of boarding passengers. If the platform section has no free capacity, the passenger moves on to the neighboring one. In a case in which the overcrowded section has two direct neighbors (i.e., the middle section), the chance that the passenger moves to one of the other two is 50%. If a vehicle's section is overcrowded, the passenger is forced to continue waiting on the platform, thereby increasing the average waiting time w per passenger. Once a vehicle has reached a passenger's final destination or transfer station, the passenger alights. Several stations have separate platforms (i.e., one per direction), so transfer times can be direction-dependent. Realistic passenger transfer times can be implemented by measuring the distances of all reasonable combinations (i.e., no transfers within the same line) at a geographic location with several stations, using the finding of Weidmann (1994) that the average walking speed of passengers is 1.34 m/s, with a deviation of $\pm 19\%$. The model uses a *triangular* distribution with $\pm 20\%$ as minimum/maximum. Alighting passengers do not use space on the platform at which they arrive, and there are no limitations on concurrent transfer operations. Due to the structural individuality of stations, some of which also serve as underpasses, modeling each station's unique entries to platforms is beyond the scope of this study.

Vehicles (in our case, subway trains) are released from the lines' end-of-line stations, in accordance with the lines' respective current headway setting, during their period of operation (from 04:30 until 01:00). Capacity depends on the vehicle type. The old U type will be replaced soon, so we focus on the remaining two. The U1 to U4 lines are served by the V type, which holds up to 878 people. The U6 line is served by type T, which holds up to 776 people. However, 100% vehicle utilization is highly undesirable and unrealistic, because at each stop, some people would have to make room and even temporarily get out to allow alighting passengers to leave. Furthermore, passengers might carry a bag, luggage, baby carriage, or bike with them. To account for these considerations, we reduce vehicle type capacity by around 20%, to 702 and 621 passengers, respectively. Their top speed is 80 km/h, and their average speeds during operation are 33.0 km/h (V type) and 30.6 km/h (T type). The *vehicle travel time* refers to the time difference between a vehicle's arrival time at two consecutive stations, so it includes the dwelling time at the first station. *Dwelling time* refers to the time difference between a vehicle's arrival at a station and its departure. It includes *boarding and alighting time*, defined as the time difference between a standing vehicle opening its doors, thereby allowing aboard passengers to alight and waiting passengers to board the vehicle, and closing them again. A comprehensive statistical analysis of vehicles' station arrival times (about 2500 samples each) reveals, that vehicle travel time depends not on time but rather on direction. Conventional wisdom might suggest that the vehicle travel time suffers from increased passenger volume at peak hours (i.e., increased dwelling time), but instead, whenever the dwelling time is longer than expected, the driver compensates by increasing the vehicle's speed. The top speed of 80 km/h represents a hard limit though. However, there seems to be enough buffer time within the schedule and enough restraint among passengers not to cause non-compensable delays. Most vehicle travel times appear almost normally distributed, with a longer tail on the right side, so we decided to use a *log-normal* distribution. Visual goodness-of-fit tests (Q–Q-plots and density-histogram plots) conducted in R provided strong support for this choice. A more detailed analysis can be found in Schmaranzer et al. (2019) and Schmaranzer et al. (2018). Once a vehicle has reached one end of a line, it remains in the station for 0.41 min (± 0.02), accounting for about 25 s of dwelling time. Thereafter, the productive fleet mileage m of the current solution increases by the respective line's length,

and the vehicle has to perform a turning maneuver. At some end-of-line stations, it is routine to turn vehicles after 20:30 simply by crossing over to the other direction's rail, prior to the arrival at the last station, to save time and vehicles. However, in most cases—depending on the infrastructure of the respective end-of-line station—a turning maneuver takes 4–8 min (± 0.34 to ± 1.00). We employ *triangular* distribution for both.

We use key performance indicators to validate the model: vehicle cycle time (i.e., vehicle travel time from one end of line station to another) and passenger-based ones (i.e., count data) at crossing stations. The deviation from the respective target values was low and approved of by the Viennese transportation provider.

This discrete event simulation model of the Viennese subway system could be used for other urban mass transit systems as well. The biggest obstacle would probably be devising the hourly origin-destination-matrices. Data about platform capacity, passenger distribution, vehicle transfer times, and turning maneuver times would be difficult to ascertain without the respective transportation provider's support. The network itself (i.e., lines and their stations) and data on the vehicle capacity can be easily be found on the internet.

Planned and unplanned service disruptions also occur in complex real urban mass rapid transit systems. Some network variants (i.e., except the full variant $l = 5$), which we introduce in Sect. 4.1, could be seen as the former (i.e., planned disruption of service on one or several lines). Unplanned disruptions are not part of our model. For studies of disruption management in general and recovery models on railway networks in particular, the reader is referred to Yu and Qi (2004) and Cacchiani et al. (2014), respectively.

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

3.2 Two-phase algorithm

Our two-phase algorithm starts by applying a weighted sum-based optimization approach to generate solutions, which then serve as starting points for an adaptation of the *multi-directional local search algorithm (MD-LS)*, as in Tricoire (2012). Such a two-phase optimization concept has already been successfully applied to combinatorial optimization problems, like the dial-a-ride (Parragh et al. 2009) or vehicle routing problem (Matl et al. 2019).

In our case, the *single-objective covariance matrix adaptation evolutionary strategy (SO-CMA-ES)* is used in the first phase (Hansen and Ostermeier 2001). Unlike genetic algorithms (Holland 1975), the SO-CMA-ES does not rely on crossover operators but creates new offspring by means of a sophisticated sampling approach (Sect. 4.2). The weight φ required for the single-objective function (Eq. 3) is set from zero to one in steps of 0.1, resulting in eleven start points, who are next handed over to the second phase. The run time t budget is split equally between both phases, so using a varying number of start points was not possible.

In the second phase, we use our *multi-directional local search (MD-LS)*, which requires the three parameters listed in Table 3. In Fig. 5 we provide a brief example and explain the basic idea of this scheme (including the aforementioned parameters). In the first stage (Fig. 5a), a set F consisting of either start points from phase one or the preceding iteration—in which case it is already a non-dominated front (i.e., Pareto front approximation)—is required. We

Table 3 Parameters of the multi-directional local search

a	Attempted moves per selected point
b	Step size
c	Selection distance

Fig. 5 Example: second phase (multi-directional local search)

use the *selection distance* parameter c (e.g., 0.1) to decide which of the already evaluated solutions in F should be added to the new set V , which only contains solutions selected for the next step (Fig. 5b). The first and last members in the sorted set F are always selected for set V ; the others only get selected if their distance to the preceding selection is at least c . Next, up to 15 new neighboring solutions based on the current selected solution v are created (Fig. 5c). The parameter *step size* b is applied to v , thereby creating new neighboring solutions. The mileage objective (Eq. 1) can be evaluated deterministically (as stated in Sect. 2.2), so we can quickly evaluate if a new neighbor is within reasonable bounds. In the third stage, we try to create up to 15 new neighbors whose objective value Z_1 lies between a search area that is restricted on the y-axis. The number of tries for this stage is limited by 100, not to be confused with the number of attempted moves a per selected point. The lower and upper bounds are defined by the selection distance parameter $c \pm 5\%$. Finally, the new neighbors are evaluated with the simulation model (Fig. 5d). Because we now have both objective values

(i.e., Z_1 and Z_2) for all neighbors, we can check whether the solutions are within the new search area, which is the overlap of the former search area on the y-axis plus a new one on the x-axis that uses its neighboring selected points' objective functions' values as bounds. In case the current v has only one neighbor (i.e., located at one of either ends), we use its current position and (0,1) or (1,0)—depending on the objective—as bounds. The best solution within the search area is then used as new basis v . The number of attempted moves per v is limited by the parameter a .

Algorithm 1 gives a more detailed explanation of our approach. It takes a set F containing potential start points. The best solutions collector Y is initialized first. The outer loop iterates

Algorithm 1 Multi-directional local search

input: F /*a set F containing potential start points*/

repeat

$Y \leftarrow \emptyset$ /*initialize empty set Y which collects the best solutions*/

for o in O **do**

Sort(F, o) /*sort set F by current objective o */

if $|F| \leq 11$ **then**

$V \leftarrow F$ /*in case there are 11 or less start points in V use all*/

else

$V \leftarrow \text{SelectStartPoints}(F, o)$ /*see Algorithm 2*/

end if

for v in V **do**

for $i \leftarrow 1$ to a **do**

$N \leftarrow \emptyset$ /*initialize empty set N of neighbors*/

$j \leftarrow 1$ /*set counter j */

$k \leftarrow 1$ /*set counter k which ensures a stop after 100 tries*/

while $j \leq 15$ **and** $k \leq 100$ **do**

$n \leftarrow \text{CreateNewNeighbor}(v)$ /*see Algorithm 3*/

PerformNumericalEvaluation(n) /*calculate estimate for Z_1 */

if n is within mileage bounds **then**

$N \leftarrow N \cup \{n\}$ **and** $j \leftarrow j + 1$ /*add to list of neighbors*/

end if

$k \leftarrow k + 1$

end while

for n in N **do**

PerformSimulationEvaluation(n) /*see Algorithm 4*/

if n is not within both bounds **then**

$N \leftarrow N \setminus \{n\}$ /*remove unsuitable solutions*/

end if

end for

Sort(N, o) /*sort set N by current objective o */

if $N \neq \emptyset$ **and** $\text{GetCurrentZ}(N.\text{first}(), o) < \text{GetCurrentZ}(v, o)$ **then**

$v \leftarrow N.\text{first}()$ /*update start point v in case a better one was found*/

else

$v \leftarrow v$ /*reuse old start point v */

end if

if $\text{GetLastElement}(Y) \neq v$ **then**

$Y \leftarrow Y \cup \{v\}$ /*add newly found best solution v to set Y */

end if

end for

end for

$F \leftarrow F \cup Y$

$F \leftarrow \text{PerformNonDominatedSorting}(F)$

until stopping criterion (run time t) is met

return F /*return non-dominated front*/

though all objectives in O , in our case Z_1 and then Z_2 . Next, F gets sorted by the current objective o . In the very first iteration, we only got eleven evaluated solutions in F (i.e., those created by the first phase). For this and the unlikely case that later, due to non-dominated sorting, there are eleven or fewer evaluated solutions in F , we select all of them (i.e., $V \leftarrow F$). Otherwise, the function “SelectStartPoints” (Algorithm 2) chooses which ones to add to V . It always adds the first and thereby best solution (with regard to the current objective o) to V . Thereafter, it ensures that the next additions keep a certain distance by using the parameter selection distance c as a measure. The last and worst solution for the current objective o also gets selected (i.e., added to V). We then iterate through the selected ones V and try to create up to 15 new neighbors using the current v as the basis.

The function “CreateNewNeighbor” (Algorithm 3) works as follows: Because the basic idea is to only change only part of the original solution v , we first iterate through every line of the solution vector. Each line has a 50% chance of being selected for small alterations. If a line is selected, a certain contiguous part of the solution vector is then chosen at random. The starting point of a potential alteration is marked by q , and the end e is another uniformly

Algorithm 2 Function SelectStartPoints

input: F, o /*sorted set F containing all potential start points and current objective o */
 $V \leftarrow F.first()$ /*initialize set V and add first element in F to it*/
for $i \leftarrow 2$ to $|F|$ **do**
 if $GetCurrentZ(F[i], o) - GetCurrentZ(V.last(), o) \geq c$ **then**
 $V \leftarrow V \cup F[i]$ /*use selection distance c (Table 3) to choose which ones to add*/
 end if
end for
if $V.last() \neq F.last()$ **then**
 $V \leftarrow V \cup F.last()$ /*ensure last element in F is also in V */
end if
return V /*return set of selected start points*/

Algorithm 3 Function CreateNewNeighbor

input: v, o /*solution v and current objective o */
 $n \leftarrow v$ /*create a copy of solution v */
 $k \leftarrow 0$ /*initialize a counter which ensures returned $n \neq v$ */
while $k = 0$ **do**
 for $i \leftarrow 1$ to l **do**
 if $uniform([0,1]) < 0.5$ **then**
 $q \leftarrow uniform(\{0, \dots, d\})$ /*set a random start point of the chosen line*/
 $e \leftarrow uniform(\{q, \dots, d\})$ /*set a random end point of the chosen line*/
 for $j \leftarrow q$ to e **do**
 $h \leftarrow uniform(\{0, \dots, 3\})$ /*how often the step size b is going to be applied*/
 if $o = Z_1$ **then**
 $n[j] \leftarrow n[j] + (b \cdot h)$ /*set wider headway*/
 else
 $n[j] \leftarrow n[j] - (b \cdot h)$ /*set tighter headway*/
 end if
 $k \leftarrow k + h$ /*increment counter k , ensuring changes where made*/
 end for
 end if
 end for
end while
return n /*return new neighbor n */

distributed point between q and d (the latter being the number of decision variables per line—see Table 2). The step size b is then applied between zero and two times (depending on the uniformly distributed integer h) to each position in the aforementioned random subsequence of the solution vector. It depends on the current objective o whether those positions' values increase or decrease. Counter k ensures that the new neighbor n is not equal to original solution v .

As described in the small example (Fig. 5c), we try to create 15 new neighbors within mileage bounds. If, after 100 trials, we have not succeeded in creating those 15 neighbors, we stop the neighborhood generation procedure. Because Z_1 is not subject to randomness, a relatively accurate value can be calculated without using the simulation model ($\pm 1\%$). To get an accurate number for Z_1 and the missing value for Z_2 , we perform a simulation-based evaluation of the new neighbors and remove those whose objectives are not within both bounds (i.e., overlapping search area) from the neighborhood N .

Finally, the neighboring solutions N get sorted according to the current objective; if the first solution in it is better than the one from which it originates, it becomes the newly selected starting point v . In each iteration, a temporary set Y stores the best solutions. Provided that it is not identical to the last element in Y , it is added to Y . It depends on the parameter a (Table 3) how often this process goes on, before continuing to the next v . After a whole iteration through all objectives O is finished, the solutions of set Y are added to the initial set F . We then perform a non-dominated sorting of the combined set, thus creating a new Pareto front approximation F that serves as input for the next iteration. The stopping criterion is the run time limit t , which depends on the complexity of the respective test instance.

4 Computational experiment setup

As stated in Sect. 2.2, we apply a normalization and scalarization approach. Table 4 contains the minimum and maximum values for *productive fleet mileage* and *average waiting time per passenger* required for the objective functions (see Eqs. 1 and 2 as well as Table 2). The technically lowest possible headway is 1.5 min, so obtaining the maximum fleet mileage $m_{\max}$ and optimal average waiting time w_{opt} is straightforward: By using the aforementioned headway value of 1.5 min for all decision variables, these extreme values could be derived. Since the average waiting time is stochastic, we used 50 replications to ensure statistical significance. Because up to 37% of passengers (depending on the network variant) perform one or two transfer operations, the optimum average waiting time is significantly higher than the expected average waiting time per waiting process (0.75 min). Due to the capacity constraint (Sect. 2.2), the lowest feasible fleet mileage $m_{\min}^*$ and the corresponding highest average waiting time $w_{\max}^*$ were more difficult to obtain: The weight φ used in Eq. 3 was set to one (i.e., only mileage minimization/optimization). We used 2-day long (i.e., 48 h) parallel optimization runs (one core for the optimizer, 15 for the simulation model) with the covariance matrix adaptation evolution strategy CMA-ES (Sect. 3.2), and used 21 decision variables d per line l (i.e., network variant), to determine the lowest feasible fleet mileage $m_{\min}^*$. To speed up optimization, and because of the fleet mileage being deterministic, three replications were used to increase the likelihood of the optimized solution being feasible. Thereafter, we ran 50 replications on the found solutions (one per network variant) to get an accurate value for the highest average waiting time $w_{\max}^*$ and to ensure the respective solution's feasibility.

As mentioned in Sect. 2.2, there are several stochastic elements, so replications are required to account for statistical significance. We employ a varying number of replications, with a

Table 4 Passenger data, m_{min}^* , m_{max} , w_{opt} and w_{max}^* per network variant

No. of lines considered l	Network length (km)	Passenger volume		Transferring passengers (%)	Fleet mileage (km)		$\varnothing$ waiting time (min)	
		(million)	(%)		m_{min}^*	m_{max}	w_{opt}	w_{max}^*
5	78.5	1.37	100	37	20,563.75	128,738.36	1.0496	7.5678
4	61.6	1.12	82	31	15,409.31	101,094.52	1.0061	7.3869
3	44.3	0.82	60	23	10,415.11	72,658.56	0.9464	7.2957
2	27.9	0.57	42	17	6287.78	45,826.52	0.9007	7.0751

minimum of three and a maximum of 50. The sequential evaluation process terminates once a 99.9% confidence interval with a relative error of one percent has been constructed. Algorithm 4 contains the function used for this purpose. The procedure works as described by Law (2013). The criteria being (as depicted in Algorithm 4) the mean waiting time w per passenger (Z_2) for all multi-objective algorithms and the multi-directional local search (i.e., the second phase). For the SO-CMA-ES (first phase) it is Z (Eq. 3). If a platform's capacity is exceeded (Sect. 2.2), the simulation is terminated immediately, the solution is deemed infeasible, and, to save time, no (more) replications are performed.

Algorithm 4 Function PerformSimulationEvaluation

```

input:  $n$  /*solution  $n$ */
 $r \leftarrow false$  /*initialize boolean stopping criteria*/
 $i \leftarrow 1$  /*initialize replication counter*/
while  $r = false$  or  $i \leq 50$  do
   $(Z_1^i, Z_2^i) \leftarrow RunSimulation(n)$  /*perform simulation run*/
  if  $n$  is infeasible then
     $r = true$  /*abort simulation evaluation in case solution is infeasible*/
  else
    if  $i \geq 3$  then
       $\delta = t_{i-1, 0.9995} \frac{StDev(Z_2)}{\sqrt{i}}$  /*calc. confidence interval using  $t$  distribution*/
      if  $\frac{\delta}{|Z_2^i|} \leq 0.01$  then
         $r \leftarrow true$  /*stop if 99.9% confidence interval with 1% relative error*/
      end if
    end if
  end if
   $i \leftarrow i + 1$ 
end while
  
```

The weight φ , as used in the single-objective function (Eq. 3), has an influence on the standard deviation of the objective value Z , and therefore the average number of replications varies, and there is a negative correlation with φ .

The huge number of samples (i.e., up to 1.37 million passenger movements) makes the standard deviation in average waiting time per passenger very small. Therefore, we introduce a “global denominator” that reduces the number of passenger entities and the capacities (platforms and vehicles) by a factor of ten. This step increased the standard deviation, but reduced the simulation run time significantly, by a factor of about six (0.58 instead of 3.52 s per run on an Intel i7-4770 with up to 3.9 GHz), with almost negligible inaccuracy with regard to the objective function (Eq. 2) values.

All experiments in this contribution were conducted on the Vienna Scientific Cluster 3 (VSC 2018), which is a high performance computing (HPC) cluster comprised of 2020

nodes, each one equipped with two Intel Xeon E5-2650v2 processors (2.6 GHz, eight cores) and at least 64 GB RAM.

4.1 Real-world test instances

The solution method described in Sect. 3 is applied to 16 different real-world test instances, based on the Viennese subway system, which serve two purposes: First, to tune (see next Sect. 4.3) the respective parameters of the applied single- and multi-objective algorithms, we needed a quick instance combination ($l \cdot d$). Second, a proper comparison of different solutions approaches necessitates multiple problem instances. The instances were created by using four different versions of the Viennese subway network l and four different numbers of decision variables d per line.

With regard to the *network versions*, Fig. 6a contains the whole Viennese subway network. It contains all five lines, so we refer to it as $l = 5$. The other variants ($l = 4$, $l = 3$ and $l = 2$; Fig. 6b–d, respectively) are reduced versions, created by removing one line at a time, according to each line's passenger volume (see Fig. 2).

Fig. 6 Network variants (schematic plan of the Viennese subway network)

For the *number of decision variables per line*, because the origin-destination-matrices change hourly, changing headways on an hourly basis is natural. The Viennese subway system operates from 04:45 to 01:00 (Monday–Thursday), so each line has 21 decision variables ($d = 21$). To fill and empty the lines with vehicles, their release starts at 04:30 and ends at 01:00. In the hourly variant, the first decision variable of each line applies to the time period prior to 05:00. Because the origin-destination-matrices are also hourly, 21 is the highest value used for d . Other variants are 2- and 3-h long headways, which produce eleven ($d = 11$) and seven ($d = 7$) decision variables per line, respectively. The smallest version re-uses headways by means of indices and works as such: each line has merely four decision variables ($d = 4$). 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 fourth and fifth as well as the 13th, 14th and 15th, for example, all have an index of two. So the solution's line's value at this particular index is used as the headway for the morning (07:00–09:00) and afternoon (16:00–19:00) peaks. The total number of decision variables lies between eight and 105 ($l \cdot d$), and the latter variant is the *largest* or *full real-world instance*.

Its optimization run time limit t was set to 110h (i.e., 110h of optimization on one single CPU core). The run time of the other instances was set in relation to the total number of decision variables (15min accuracy). The smallest one has a optimization run time of 13.75h. One evaluation of all 16 test instances takes about 728.75h of computation on a single CPU core. Given, that three different algorithms are used, and five independent optimization runs (not to be confused with replications) are performed, a total of 10,931.25h is required. The experiments were conducted on the VSC3 (VSC 2018), as introduced at the end of Sect. 4. To speed up the whole process, several nodes with 16 CPU cores each were used simultaneously.

4.2 Other multi-objective algorithms and solution encoding

In addition to the two-phase algorithm (SO-CMA-ES and MD-LS), introduced in Sect. 3.2, we implemented a non-dominated sorted genetic algorithm (NSGA-II) and the multi-objective covariance matrix adaptation evolutionary strategy (MO-CMA-ES) for comparison purposes.

The *NSGA-II* (Deb et al. 2002) is a genetic algorithm for multi-objective optimization. Its selection mechanism is rank-based and relies on the identification of non-dominated solutions in the population. Apart from crossover and mutation probability, two parameters, the population size and selected parents factor, need to be set. Crossover and mutation operators create new candidate solutions (i.e., offspring). We used three standard crossover operators for real-valued encodings: (1) an average crossover that calculates an average value of the parents' values at the respective position of their gene material; (2) an arithmetic crossover that randomly performs an average calculation or simply takes the value from the first parent; and (3) a blend alpha crossover (Takahashi and Kita 2001) that calculates an interval and uses it as boundary for a new random value. For each offspring to be created, one of the crossover operators is chosen at random. Thereafter, a fast non-dominated sorting procedure ranks all solutions, thereby structuring the population into several fronts.

The *MO-CMA-ES* (Igel et al. 2007) is the multi-objective version of the single-objective covariance matrix adaptation evolutionary strategy (SO-CMA-ES), which we used in the first phase of our two-phase optimization scheme (Sect. 3.2). Unlike the NSGA-II, which

relies on crossover and mutation operators, it creates new candidate solutions by means of a sophisticated sampling approach. However, it is similar to the NSGA-II in terms of elitism and selection based on non-dominated sorting.

The algorithms' initial populations are based on the currently employed, real-world headways. The base solution is the very first individual in the initial population, and the remaining ones are generated by sampling from a normal distribution with the currently employed headways as mean and 20% standard deviation.

Schmaranzer et al. (2019) study four different solution encoding variants (including continuous and discrete values) and find that the SO-CMA-ES achieved the best results, in less time, when using continuous factor encoding. This encoding uses factors instead of values, applied to the currently employed headways. If, for example, the initial headway is 5.0 min and the factor (i.e., decision variable) is 1.1, the resulting headway is 5.5 min. The desired lower and upper bounds for the final headways are 1.5 and 20 min, so the bounds for the factors have to be set accordingly.

As for software, we used several libraries from [HeuristicLab 3.3.15](#) (Wagner et al. 2014), which is a metaheuristics framework developed in C#.

4.3 Algorithm parameter tuning

Algorithms have various parameters that must be *tuned* to fit the problem. We defined reasonable values for the parameters and ran full factorial experiments, as follows: the population size (i.e., front size) was set to 50, 75, 100, 150, and 200. Because the SO/MO-CMA-ES usually tend toward lower population sizes, the following values were tested: 35, 50, 65, 80, and 100. In the NSGA-II, the crossover and mutation probabilities were set to 90% and 100%, and to 5%, 10%, 15%, 20%, 30%, and 40%, respectively. The selected parents were set to 1.5 and 2.0 times the population size. The initial σ in both CMA-ES was set as a fraction of the parameter range. Thereby, its resulting value depends on the solution encoding's bounds. Values of 1/8, 1/6, and 1/4 were tested. For the remaining SO-CMA-ES parameters, we tested 0, 50, 100, 150, 200, 300, and 500 initial iterations. The μ parameter was set to "NULL", 1, 5, and 10. Finally, for the parameters of MD-LS, the step size was set to 0.005, 0.010, 0.025, 0.050, and 0.1. The number of attempted moves per points was set to 3, 5, 7, 10, and 15. The selection distance values of 0.05, 0.10, and 0.15 were tested.

The instance used for tuning has only two lines ($l = 2$) and 21 parameters ($d = 21$) per line (Sect. 4). Therefore, this instance has 42 decision variables ($l \cdot d$) in total and only one crossing station (Fig. 6d). This combination of the network l and number of decision variables d per line offers the best combination of low run time (due to the significantly reduced passenger volume) but still a sufficiently high number of headways to be set. Table 5 contains the tuned parameter values for all algorithms. Note that the population size does not vary that much in most cases. Higher population sizes (up to 200 here and up to 300 in Schmaranzer et al. 2018) have been tested, but lower ones lead to better results, likely due to the long evaluation time. Apart from the population size (or front size, in case of the MD-LS), the algorithms have different parameters. If an algorithm does not feature a certain parameter, it is marked by "–".

The NSGA-II's rather high mutation probability is likely due to the already decent solution currently being employed, on which the initial solution is based. Another explanation could be the low diversity within the initial population.

Table 5 Tuned parameters for all algorithms

Parameter designation	NSGA-II	MO-CMA-ES	SO-CMA-ES (1st phase)	MD-LS (2nd phase)
Population/front size	50	50	50	50
Crossover probability	90%	–	–	–
Mutation probability	40%	–	–	–
Selected parents	200	–	–	–
Initial σ (fraction of parameter range)	–	1/8	1/8	–
Initial iterations	–	–	150	–
μ	–	–	Null	–
Attempted moves a per selected point	–	–	–	10
Step size b	–	–	–	0.05
Selection distance c	–	–	–	0.05

5 Computational results

This section is structured as follows: We examine the results of the first phase from our algorithm (Sect. 5.1), then present the results and summarize them in tabular form (Sect. 5.2). Next, Sect. 5.3 compares the resulting best Pareto front approximations. Section 5.4 details the real-world instance.

5.1 Analysis of the first phase

The result of the first phase are eleven start points that serve as input for the second phase. Figure 7 contains four of 16 instances' starting points; the remaining twelve can be found in Appendix A.1. They contain one of each network (i.e., number of lines l), and one of each decision variables per line d variants. As mentioned at the end of Sect. 4.1, five independent and reproducible optimization runs were performed. The low standard deviation prompted us to plot only the ones that led to the best result (i.e., highest hypervolume) in the second phase. We decided to use normalized values because we compare different network variants l .

Figure 7a depicts the smallest instance with a total of eight decision variables. The best solution in terms of quality of fleet mileage is very close to the optimum; the same holds for the other instances with more decision variables (Fig. 7b–d). However, the quality of average waiting time per passenger deteriorates from 0.0035, 0.0037, 0.0889, and finally to 0.1324. This deterioration becomes more apparent when looking at the corresponding fleet mileage qualities: 0.7332, 0.6257, 0.5340, and 0.4376. One reason for the better solutions on one end is the aforementioned (Sect. 2.2) influence of the weight φ on the required number of replications in the single-objective case (i.e., SO-CMA-ES only). The lower the weight φ (i.e., higher priority on average waiting time per passenger), the higher the number of replications. We know from previous works (Schmaranzer et al. 2018, 2019) that the number of evaluated solution with a weight of $\varphi = 0.25$ decreases by almost 30% and increases by about 16% with a weight of $\varphi = 0.75$, when compared with the equally balanced $\varphi = 0.50$.

Fig. 7 Excerpt: Eleven start positions from phase one (best out of five runs)

Apart from managing fewer evaluations due to the influence of the weight φ on Z subject to randomness (SO-CMA-ES; single-objective case only), we must also consider that average waiting time per passenger optimization is harder than fleet mileage optimization. The average waiting time per passenger is time-dependent, because so is the passenger volume (Fig. 2). So a slightly tighter headway during a peak hour may lead to a better result than a much tighter headway in an off-peak hour. Fleet mileage is not time-dependent; it simply does not matter if the headway widened during peak or off-peak hours. As long as a wider headway does not lead to infeasibility and reduces the number of vehicle releases, the resulting fleet mileage is reduced. Of course, the affected line's length has an influence on how great the savings are, but the affected position in the solution vector has no influence.

5.2 Multi-objective optimization results

Table 6 contains the average results obtained from the three algorithms, NSGA-II, MO-CMA-ES, and the two-phase algorithm, in terms of the resulting fronts' hypervolumes (Zitzler and Thiele 1999). The hypervolume measures the size of the objective space covered by an approximation set. According to Riquelme et al. (2015), it is the most commonly used performance metric in multi-objective optimization, also known as S metric, hyper-area, or Lebesgue measure. It is unary and considers accuracy, diversity, and cardinality. The required reference point was set to (1.1, 1.1). It was not possible to use the concept of the Nadir-point (Ehrgott and Tenfelde-Podehl 2003) for that purpose, because the required lexicographic optimization could not be conducted in this stochastic, simulation-based context.

In the smallest test instance (8 decision variables), the MO-CM-ES performs best. However, all three optimization schemes perform rather well. The relative difference among the three competitors is 0.98%. Up to $l = 4$, the test instances with the smallest number of decision variables for each line ($d = 4$) is solved best by the MO-CMA-ES, followed by our proposed two-phase algorithm. The latter performs best in 13 test instances and second best in the remaining three. Considering that the NSGA-II ranks third in all 16 instances, we used its results as a baseline. When compared with the other two algorithms, it becomes apparent that the relative distance increases with the number of lines l and the number of variables per line d . The NSGA-II's performance is decreasing when comparing the four hypervolume results within a specific number of lines l . This effect can also be observed from the MO-CMA-ES results. Overall, the two-phase algorithm performed

Table 6 Tabular average results (five independent and reproducible optimization runs; best in bold; worse in italic)

No. of lines $[l]$	dec. var. $[l \cdot d]$	Run time t (h) (1 core)	Hypervolume (higher is better)			Difference to NSGA-II (%)	
			NSGA-II	MO-CMA	2-phase	MO-CMA	2-phase
2	8	13.75	<i>1.0362</i>	1.0462	1.0400	0.96	0.36
	14	27.50	<i>1.0310</i>	1.0418	1.0432	1.05	1.18
	22	27.50	<i>1.0262</i>	1.0425	1.0446	1.60	1.80
	42	55.00	<i>1.0179</i>	1.0367	1.0416	1.84	2.33
3	12	27.50	<i>1.0396</i>	1.0477	1.0449	0.78	0.51
	21	27.50	<i>1.0212</i>	1.0418	1.0419	2.02	2.03
	33	41.25	<i>1.0108</i>	1.0401	1.0452	2.90	3.41
	63	68.75	<i>0.9919</i>	1.0227	1.0470	3.10	5.56
4	16	27.50	<i>1.0173</i>	1.0397	1.0380	2.20	2.03
	28	27.50	<i>0.9760</i>	1.0229	1.0335	4.81	5.89
	44	55.00	<i>0.9743</i>	1.0271	1.0401	5.42	6.75
	84	96.25	<i>0.9535</i>	1.0132	1.0419	6.26	9.27
5	20	27.50	<i>1.0015</i>	1.0309	1.0420	2.94	4.05
	35	41.25	<i>0.9675</i>	1.0155	1.0353	4.97	7.01
	55	55.00	<i>0.9488</i>	1.0036	1.0340	5.77	8.98
	105	110.00	<i>0.9234</i>	0.9960	1.0362	7.87	12.22
Average			<i>0.9961</i>	1.0293	1.0406	3.40	4.59

4.59%, and the MO-CMA-ES 3.40%, better compared with the NSGA-II. The hypervolume in the largest test instance with 105 decision variables was considerably smaller for the NSGA-II.

5.3 Details on the best Pareto front approximations

Table 7 contains the multiplicative binary ϵ (Zitzler et al. 2003) of the MO-CMA-ES and the NSGA-II (best out of five), where our two-phase optimization scheme serves as a reference point. The ϵ indicator gives the factors by which approximation sets (MO-CMA-ES and NSGA-II) are worse than a reference set (two-phase) with respect to all objectives. Precisely speaking, the ϵ -values reported in Table 7 represent the smallest quantities by which the objective vectors in the approximation set of our two-phase approach have to be multiplied to ensure that the whole set is weakly dominated by the respective comparison set. Our two-phase optimization scheme performs rather well, and the MO-CMA-ES can handle the increasing complexity (i.e., number of lines l and resulting total number of decision variables $l \cdot d$) better than the NSGA-II.

Figures 8, 9, 10 and 11 contain the best Pareto fronts approximations (with respect to the hypervolume) for all algorithms (best out of five independent and reproducible optimization runs). As in Sect. 5.1, we included four representative plots by increasing the network l and number of decision variables per line d variants, from the smallest to the largest test instance; the remaining twelve plots can be observed in Appendix A.2.

In Fig. 8, there is not much of a difference among the three solution schemes. All cover a broad range, though the MO-CMA-ES and the two-phase algorithm cover a slightly higher spread (i.e., get closer to the optimum) in terms of fleet mileage quality (lower right corner).

Table 7 Binary multiplicative ϵ with two-phase as reference

No. of lines l	dec. var. $[l \cdot d]$	Run time t [h] (1 core)	Binary multiplicative ϵ (ref.: 2-phase)	
			MO-CMA-ES	NSGA-II
2	8	13.75	1.0282	1.0561
	14	27.50	1.0272	1.0823
	22	27.50	1.0390	1.1005
	42	55.00	1.0536	1.1981
3	12	27.50	1.0203	1.0703
	21	27.50	1.0278	1.1268
	33	41.25	1.0467	1.2503
	63	68.75	1.0768	1.3825
4	16	27.50	1.0254	1.1663
	28	27.50	1.0814	1.4542
	44	55.00	1.0700	1.5028
	84	96.25	1.1542	1.5634
5	20	27.50	1.0852	1.2835
	35	41.25	1.2104	1.5342
	55	55.00	1.2562	1.5133
	105	110.00	1.1944	1.6702
Average			1.0873	1.3097

With 21 decision variables (Fig. 9), the MO-CMA-ES, followed by the two-phase scheme, cover a wide range of the Pareto front approximation. Also, both manage to move the front closer to the origin.

In the test instance with 44 decision variables (Fig. 10), the trend continues: the MO-CMA-ES and two-phase algorithm perform best. The NSGA-II covers a far smaller spread for both objectives.

Figure 11 contains the largest test instance. Here, the two-phase optimization scheme performs best, in terms of both coverage and in moving the Pareto front approximation toward the origin. In the middle of the front, the NSGA-II manages to produce better solutions than the MO-CMA-ES. However, in terms of spread the MO-CMA-ES performs better than the NSGA-II.

A reason for the MO-CMA-ES's good performance in terms of average waiting time per passenger (Z_2) is that it does not use bounds in a traditional manner Igel et al. (2007). Whenever a newly created solution violates the bounds, it does not correct it but evaluates

Fig. 8 Eight decision variables ($l = 2$; $d = 4$)

Fig. 9 21 decision variables ($l = 3$; $d = 7$)

Fig. 10 44 decision variables
($l = 4$; $d = 11$)

Fig. 11 105 decision variables
($l = 5$; $d = 21$)

the next feasible one and adds a tiny penalty to its quality, we only used the quality values without any penalties. This aspect, in conjunction with the continuous factor encoding, creates the following effect: when the MO-CMA-ES uses a negative factor, the resulting headway is negative, such that it gets replaced by the lower bound of 1.5 min, which is also the technically lowest possible headway. The end result is good quality for the waiting time's end of the front.

Moshaiov and Abramovich (2014) use the NSGA-II and MO-CMA-ES for the evolution of multi-objective neuro-controllers. They also conclude that the advantage of the NSGA-II (i.e., better solutions compared with the MO-CMA-ES) is restricted to part of the front, whereas the MO-CMA-ES is superior over a large part of the front. Because the results from the first phase in our two-phase optimization scheme (Sect. 5.1) do not provide good results in terms of Z_2 (i.e., waiting time) for test instances with a high number of decision variables, our multi-directional local search managed to

improve that end of the final front quite a bit (e.g., 0.1324–0.0037 for the largest instance).

5.4 Detailed results of the real-world instance

Practitioners are often interested in multiple solutions from which they can choose. Furthermore, extreme solutions that lead to, for example, 50% more fleet mileage with similar increases in fleet size and drivers create real-world implementation issues, especially in the short term. A large order of new vehicles requires time and preparations by manufacturers, as does the process for recruiting and training of new drivers.

Table 8 contains some results on solutions that are close to the currently employed headways, which result in 45,943 km fleet mileage and 2.69 min average waiting time per passenger. To ensure and further improve the solutions' accuracy, all were re-evaluated using 50 replications. The one in bold, for example, results in 44,579 km and 2.67 min. It actually offers both a cost reduction of about 3% (less fleet mileage) and service quality improvement of 0.7%, compared with the currently employed setting.

As mentioned, the average invehicle time and both average transfer times do not vary. The average total travel time thus is driven by the average waiting time per passenger, which justifies our choice to use it as a service level indicator.

Table 8 Selected solutions from best Pareto front approx. (two-phase algorithm)

Fleet mileage (km)	Average passenger times [minutes] (all)				Average transfer time [minutes] (transferring passengers only)
	Waiting	Invehicle	Transfer	Travel	
37,777	3.15	9.05	0.71	12.91	2.31
39,806	3.00	9.05	0.71	12.77	2.31
40,724	2.93	9.05	0.71	12.70	2.31
42,145	2.83	9.05	0.71	12.59	2.31
43,517	2.73	9.05	0.71	12.50	2.31
44,579	2.67	9.05	0.71	12.44	2.31
46,164	2.59	9.06	0.71	12.36	2.31
48,860	2.45	9.06	0.71	12.22	2.31
51,107	2.34	9.06	0.71	12.12	2.31
52,930	2.26	9.06	0.71	12.03	2.31
54,803	2.19	9.06	0.71	11.96	2.31

6 Conclusion and perspectives

We developed a two-phase multi-objective optimization scheme, comprised of a single-objective CMA-ES and a multi-directional local search, and compared it with two well-known multi-objective population-based algorithms (NSGA-II and MO-CMA-ES). The algorithms were applied to a multi-objective simulation-based headway optimization problem for complex public mass rapid transit systems. The first phase of our algorithm uses a single-objective algorithm (SO-CMA-ES), which yields eleventh start points, from which the multi-directional local search embarks on its task of creating a Pareto front approximation. Computational experience is gained from several test instances based on a real-world subway network (i.e., the Viennese mass rapid transit system). The two-phase optimization scheme performed best in 81% of the test instances. The NSGA-II's performance deteriorated with increasing test instance complexity, yet it yields better solutions than the MO-CMA-ES in a restricted part of the front, whereas the MO-CMA-ES is superior over a large spread of the front. The headways that are currently employed by the transport provider already offer a good balance between the objectives of fleet mileage and average waiting time per passenger reduction. However, one of the solutions derived by our proposed multi-objective algorithm offers both, a cost reduction of about 3% (less fleet mileage) and a service quality improvement of 0.7% (lower average waiting time per passenger).

An extension of the proposed approach to other means of public transport, like trams or buses, would offer an interesting pathway for further research. Kiefer et al. (2018) recently investigated public transport network preventive maintenance tasks. The headways for the required replacement services could be determined by our multi-objective simulation optimization scheme. Other extensions involve the implementation of planned disruptions. For the time being, the real-world system is still unaffected by this study. Possible future impacts include changes in the lines' respective hourly headways (i.e. new schedule), planning of vehicle acquisition, infrastructure alterations, and disruption management.

Acknowledgements Open access funding provided by University of Vienna. The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, and Wiener Linien GmbH & Co KG (i.e., the Viennese public transportation provider) is gratefully acknowledged.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

A Additional plots

A.1 Startpoints

Figures 12, 13, 14 and 15 contain the plots not included in Sect. 5.1 (three per network variant).

Fig. 12 $l = 2$ network variant

Fig. 13 $l = 3$ network variant

Fig. 14 $l = 4$ network variant **(a)** 16 decision variables ($d = 4$). **(b)** 28 decision variables ($d = 7$). **(c)** 84 decision variables ($d = 21$).

Fig. 15 $l = 5$ network variant **(a)** 20 decision variables ($d = 4$). **(b)** 35 decision variables ($d = 7$). **(c)** 55 decision variables ($d = 11$).

A.2 Best fronts

Figures 16, 17, 18 and 19 contain the plots not included in Sect. 5.3 (three per network variant).

Fig. 16 $l = 2$ network variant

(a) 14 decision variables ($d = 7$).

(b) 22 decision variables ($d = 11$).

(c) 42 decision variables ($d = 21$).

Fig. 17 $l = 3$ network variant **(a)** 12 decision variables ($d = 4$). **(b)** 33 decision variables ($d = 11$). **(c)** 63 decision variables ($d = 21$).

Fig. 18 $l = 4$ network variant **(a)** 16 decision variables ($d = 4$). **(b)** 28 decision variables ($d = 7$). **(c)** 84 decision variables ($d = 21$).

Fig. 19 $l = 5$ network variant **(a)** 20 decision variables ($d = 4$). **(b)** 35 decision variables ($d = 7$). **(c)** 55 decision variables ($d = 11$).

References

- Agard, B., Morency, C., & Trépanier, M. (2007). *Mining public transport user behaviour from smart card data*. Technical report, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT). <https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2007-42.pdf>. Accessed 27 July 2015.
- Amaran, S., Sahinidis, N. V., Sharda, B., & Bury, S. J. (2016). Simulation optimization: A review of algorithms and applications. *Annals of Operations Research*, 240(1), 351–380. <https://doi.org/10.1007/s10479-015-2019-x>.
- Bauer, D., Brändle, N., Seer, S., Ray, M., & Kitazawa, K. (2009). Measurement of pedestrian movements: A comparative study on various existing systems. In H. Timmermans (Ed.), *Pedestrian behavior: Models, data collection and applications*. Emerald: Bingley.
- Bovy, P. H. L., & Stern, E. (1990). *Route choice: Wayfinding in transport networks. Studies in operational regional science* (Vol. 9). Dordrecht: Springer. <https://doi.org/10.1007/978-94-009-0633-4>.
- Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., et al. (2014). An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63, 15–37. <https://doi.org/10.1016/j.trb.2014.01.009>.
- Ceder, A. (1984). Bus frequency determination using passenger count data. *Transportation Research Part A: General*, 18(5–6), 439–453. [https://doi.org/10.1016/0191-2607\(84\)90019-0](https://doi.org/10.1016/0191-2607(84)90019-0).
- Ceder, A. (2001). Bus timetables with even passenger loads as opposed to even headways. *Transportation Research Record: Journal of the Transportation Research Board*, 1760, 3–9. <https://doi.org/10.3141/1760-01>.
- Ceder, A., & Wilson, N. H. (1986). Bus network design. *Transportation Research Part B: Methodological*, 20(4), 331–344. [https://doi.org/10.1016/0191-2615\(86\)90047-0](https://doi.org/10.1016/0191-2615(86)90047-0).
- Chong, L., & Osorio, C. (2018). A simulation-based optimization algorithm for dynamic large-scale urban transportation problems. *Transportation Science*, 52(3), 637–656. <https://doi.org/10.1287/trsc.2016.0717>.
- Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2), 182–197. <https://doi.org/10.1109/4235.996017>.
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271. <https://doi.org/10.1007/BF01386390>.
- Dollevet, T., Huisman, D., Kroon, L., Schmidt, M., & Schöbel, A. (2015). Delay management including capacities of stations. *Transportation Science*, 49(2), 185–203. <https://doi.org/10.1287/trsc.2013.0506>.
- Ehrgott, M., & Tenfelde-Podehl, D. (2003). Computation of ideal and Nadir values and implications for their use in MCDM methods. *European Journal of Operational Research*, 151(1), 119–139. [https://doi.org/10.1016/S0377-2217\(02\)00595-7](https://doi.org/10.1016/S0377-2217(02)00595-7). Accessed 23 June 2016.
- Farahani, R. Z., Miandoabchi, E., Szeto, W., & Rashidi, H. (2013). A review of urban transportation network design problems. *European Journal of Operational Research*, 229(2), 281–302. <https://doi.org/10.1016/j.ejor.2013.01.001>.
- Frejinger, E. (2008). *Route choice analysis: Data, models, algorithms and applications*. PhD-Thesis, École polytechnique fédérale de Lausanne, Lausanne, https://www.researchgate.net/profile/Emma-Frejinger/publication/37454429_Route_choice_analysis_data_models_algorithms_and_applications/links/0046352922b197b0d5000000.pdf. Accessed 23 June 2016.
- Friedrich, M., Immisch, K., Jehlicka, P., Otterstätter, T., & Schlaich, J. (2010). Generating origin–destination matrices from mobile phone trajectories. *Transportation Research Record: Journal of the Transportation Research Board*, 2196, 93–101. <https://doi.org/10.3141/2196-10>.
- Fu, M. C. (2002). Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14(3), 192–215. <https://doi.org/10.1287/ijoc.14.3.192.113>.
- Furth, P. G., & Wilson, N. H. (1981). Setting frequencies on bus routes: Theory and practice. *Transportation Research Record*, 818, 1–7.
- Guihaire, V., & Hao, J. K. (2008). Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice*, 42(10), 1251–1273. <https://doi.org/10.1016/j.tra.2008.03.011>.
- Han, A. F., & Wilson, N. H. (1982). The allocation of buses in heavily utilized networks with overlapping routes. *Transportation Research Part B: Methodological*, 16(3), 221–232. [https://doi.org/10.1016/0191-2615\(82\)90025-X](https://doi.org/10.1016/0191-2615(82)90025-X).
- Hanika, A. (2018). Zukünftige Bevölkerungsentwicklung Österreichs und der Bundesländer 2017 bis 2080 (Teil 2). *Statistische Nachrichten*, 3, 218–225.
- Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2), 159–195. <https://doi.org/10.1162/106365601750190398>.

- Herbon, A., & Hadas, Y. (2015). Determining optimal frequency and vehicle capacity for public transit routes: A generalized newsvendor model. *Transportation Research Part B: Methodological*, 71, 85–99. <https://doi.org/10.1016/j.trb.2014.10.007>.
- Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence* (1st ed.). Cambridge, MA: MIT Press. Complex adaptive systems.
- Huang, Z., Ren, G., & Liu, H. (2013). Optimizing bus frequencies under uncertain demand: Case study of the transit network in a developing city. *Mathematical Problems in Engineering*, 2013, 1–10. <https://doi.org/10.1155/2013/375084>.
- Ibarra-Rojas, O., Delgado, F., Giesen, R., & Muñoz, J. (2015). Planning, operation, and control of bus transport systems: A literature review. *Transportation Research Part B: Methodological*, 77, 38–75. <https://doi.org/10.1016/j.trb.2015.03.002>.
- Igel, C., Hansen, N., & Roth, S. (2007). Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1), 1–28. <https://doi.org/10.1162/evco.2007.15.1.1>.
- IKK. (2017). *Matchmobile—multimodal trip chains from mobile phones*. <http://www.ikk.at/projekt/matchmobile-multimodal-trip-chains-from-mobile-phones/>. Accessed 20 June 2017.
- Jackson, J. R. (1963). Jobshop-like queueing systems. *Management Science*, 10(1), 131–142. <https://doi.org/10.1287/mnsc.10.1.131>.
- Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., & Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2, 62–72. <https://doi.org/10.1016/j.orp.2015.03.001>.
- Kiefer, A., Schilde, M., & Doerner, K. F. (2018). Scheduling of maintenance work of a large-scale tramway network. *European Journal of Operational Research*, 270(3), 1158–1170. <https://doi.org/10.1016/j.ejor.2018.04.027>.
- Law, A. M. (2013). *Simulation modeling and analysis*. Industrial engineering and management science (5th ed.). Dubuque: McGraw-Hill Education.
- Li, Y., Xu, W., & Hm, S. (2013). Expected value model for optimizing the multiple bus headways. *Applied Mathematics and Computation*, 219(11), 5849–5861. <https://doi.org/10.1016/j.amc.2012.11.098>.
- Liebchen, C. (2008). The first optimized railway timetable in practice. *Transportation Science*, 42(4), 420–435. <https://doi.org/10.1287/trsc.1080.0240>.
- Matl, P., Hartl, R. F., & Vidal, T. (2019). Leveraging single-objective heuristics to solve bi-objective problems: Heuristic box splitting and its application to vehicle routing. *Networks*. <https://doi.org/10.1002/net.21876>.
- Mohaymany, A. S., & Amiripour, S. M. (2009). Creating bus timetables under stochastic demand. *International Journal of Industrial Engineering & Production Research*, 20(3), 83–91.
- Moshaiou, A., & Abramovich, O. (2014). Is MO-CMA-ES superior to NSGA-II for the evolution of multi-objective neuro-controllers? In *2014 IEEE congress on evolutionary computation (CEC)*. IEEE, Beijing, China (pp. 2809–2816). <https://doi.org/10.1109/CEC.2014.6900433>.
- Newell, G. F. (1971). Dispatching policies for a transportation route. *Transportation Science*, 5(1), 91–105. <https://doi.org/10.1287/trsc.5.1.91>.
- Niu, H., & Zhou, X. (2013). Optimizing urban rail timetable under time-dependent demand and oversaturated conditions. *Transportation Research Part C: Emerging Technologies*, 36, 212–230. <https://doi.org/10.1016/j.trc.2013.08.016>.
- Osorio, C., & Bierlaire, M. (2013). A simulation-based optimization framework for urban transportation problems. *Operations Research*, 61(6), 1333–1345. <https://doi.org/10.1287/opre.2013.1226>.
- Osorio, C., & Chong, L. (2015). A computationally efficient simulation-based optimization algorithm for large-scale urban transportation problems. *Transportation Science*, 49(3), 623–636. <https://doi.org/10.1287/trsc.2014.0550>.
- Parragh, S. N., Doerner, K. F., Hartl, R. F., & Gandibleux, X. (2009). A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. *Networks*, 54(4), 227–242. <https://doi.org/10.1002/net.20335>.
- Pelletier, M. P., Trépanier, M., & Morency, C. (2011). Smart card data use in public transit: A literature review. *Transportation Research Part C: Emerging Technologies*, 19(4), 557–568. <https://doi.org/10.1016/j.trc.2010.12.003>.
- Raveau, S., Guo, Z., Muñoz, J. C., & Wilson, N. H. (2014). A behavioural comparison of route choice on metro networks: Time, transfers, crowding, topology and socio-demographics. *Transportation Research Part A: Policy and Practice*, 66, 185–195. <https://doi.org/10.1016/j.tra.2014.05.010>.
- Riquelme, N., Von Lucken, C., & Baran, B. (2015). Performance metrics in multi-objective optimization. In *2015 Latin American computing conference (CLEI)*, IEEE, Arequipa, Peru (pp. 1–11). <https://doi.org/10.1109/CLEI.2015.7360024>.

- Roth, C., Kang, S. M., Batty, M., & Barthelemy, M. (2012). A long-time limit for world subway networks. *Journal of The Royal Society Interface*, 9(75), 2540–2550. <https://doi.org/10.1098/rsif.2012.0259>.
- Ruano, E., Cobos, C., & Torres-Jimenez, J. (2017). Transit network frequencies-setting problem solved using a new multi-objective global-best harmony search algorithm and discrete event simulation. In O. Pichardo-Lagunas & S. Miranda-Jiménez (Eds.), *Advances in soft computing* (Vol. 10062, pp. 341–352). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-62428-0_27.
- Salzborn, F. J. M. (1972). Optimum bus scheduling. *Transportation Science*, 6(2), 137–148. <https://doi.org/10.1287/trsc.6.2.137>.
- Schmaranzer, D., Braune, R., & Doerner, K. F. (2016). A discrete event simulation model of the Viennese subway system for decision support and strategic planning. In T. Roeder, P. Frazier, R. Szechtmann, E. Zhou, T. Huschka, & Chick, S. (Eds.), *Proceedings of the 2016 Winter simulation conference*, Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ (pp. 2406–2417). <https://doi.org/10.1109/WSC.2016.7822280>, <http://www.informs-sim.org/wsc16papers/210.pdf>.
- Schmaranzer, D., Braune, R., & Doerner, K. F. (2018). Simulation-based headway optimization for a subway network: A performance comparison of population-based algorithms. In M. Rabe, A. J. Angel, N. Mustafee, A. Skoogh, S. Jain, & B. Johansson (Eds.), *Proceedings of the 2018 Winter simulation conference*, Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ (pp. 1957–1968). <https://doi.org/10.1109/WSC.2018.8632362>, <https://www.informs-sim.org/wsc18papers/includes/files/164.pdf>.
- Schmaranzer, D., Braune, R., & Doerner, K. F. (2019). Population-based simulation optimization for urban mass rapid transit networks. *Flexible Services and Manufacturing Journal*, <https://doi.org/10.1007/s10696-019-09352-9>.
- Schéele, S. (1980). A supply model for public transit services. *Transportation Research Part B: Methodological*, 14(1–2), 133–146. [https://doi.org/10.1016/0191-2615\(80\)90039-9](https://doi.org/10.1016/0191-2615(80)90039-9).
- Shrivastava, P., & Dhingra, S. L. (2002). Development of coordinated schedules using genetic algorithms. *Journal of Transportation Engineering*, 128(1), 89–96. [https://doi.org/10.1061/\(ASCE\)0733-947X\(2002\)128:1\(89\)](https://doi.org/10.1061/(ASCE)0733-947X(2002)128:1(89)).
- Shrivastava, P., Dhingra, S. L., & Gundaliya, P. J. (2002). Application of genetic algorithm for scheduling and schedule coordination problems. *Journal of Advanced Transportation*, 36(1), 23–41. <https://doi.org/10.1002/atr.5670360103>.
- Statistik Austria. (2017). Download Bevölkerungsprognosen (ausführliche Tabellen Wien). <http://www.statistik.at/wcm/idc/idcplg?IdcService=GET&RevisionSelectionMethod=LatestReleased&dDocName=027327>. Accessed 3 July 2018.
- Sun, L., Jin, J. G., Lee, D. H., Axhausen, K. W., & Erath, A. (2014). Demand-driven timetable design for metro services. *Transportation Research Part C: Emerging Technologies*, 46, 284–299. <https://doi.org/10.1016/j.trc.2014.06.003>.
- Takahashi, M., & Kita, H. (2001). A crossover operator using independent component analysis for real-coded genetic algorithms. *Proceedings of the 2001 Congress on Evolutionary Computation, IEEE*, 1, 643–649. <https://doi.org/10.1109/CEC.2001.934452>.
- Tricoire, F. (2012). Multi-directional local search. *Computers & Operations Research*, 39(12), 3089–3101. <https://doi.org/10.1016/j.cor.2012.03.010>.
- United Nations. (2018). *Download annual percentage of population at mid-year residing in urban areas by region, subregion and country, 1950–2050*. https://esa.un.org/unpd/wup/Download/Files/WUP2018-F21-Proportion_Urban_Annual.xls. Accessed 3 July 2018.
- VSC. (2018). Vienne scientific cluster. <http://vsc.ac.at/systems/vsc-3/>. Accessed 3 Apr 2018.
- Vázquez-Abad, F. J., & Zubieta, L. (2005). Ghost simulation model for the optimization of an urban subway system. *Discrete Event Dynamic Systems*, 15(3), 207–235. <https://doi.org/10.1007/s10626-005-2865-9>.
- Wagner, S., Kronberger, G., Beham, A., Kommenda, M., Scheibenpflug, A., Pitzer, E., et al. (2014). Architecture and design of the HeuristicLab optimization environment. In R. Klempous, J. Nikodem, W. Jacak, & Z. Chaczko (Eds.), *Advanced methods and applications in computational intelligence, Topics in intelligent engineering and informatics* (Vol. 6, pp. 197–261). Heidelberg: Springer International Publishing. https://doi.org/10.1007/978-3-319-01436-4_10.
- Weidmann, U. (1994). Der Fahrgastwechsel im öffentlichen Personenverkehr. Dissertation, ETH Zürich. <https://doi.org/10.3929/ethz-a-000948819>.
- Wu, J., Song, R., Wang, Y., Chen, F., & Li, S. (2015). Modeling the coordinated operation between bus rapid transit and bus. *Mathematical Problems in Engineering*, 2015, 1–7. <https://doi.org/10.1155/2015/709389>.
- Yu, B., Yang, Z., Sun, X., Yao, B., Zeng, Q., & Jeppesen, E. (2011). Parallel genetic algorithm in bus route headway optimization. *Applied Soft Computing*, 11(8), 5081–5091. <https://doi.org/10.1016/j.asoc.2011.05.051>.

- Yu, G., & Qi, X. (2004). *Disruption management: Framework, models and applications*. Hackensack, NJ: World Scientific. oCLC: 255210733.
- Zhao, F., & Zeng, X. (2006). Optimization of transit network layout and headway with a combined genetic algorithm and simulated annealing method. *Engineering Optimization*, 38(6), 701–722. <https://doi.org/10.1080/03052150600608917>.
- Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4), 257–271. <https://doi.org/10.1109/4235.797969>.
- Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., & da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2), 117–132. <https://doi.org/10.1109/TEVC.2003.810758>.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.