Calibrating Path Choices and Train Capacities for Urban Rail Transit Simulation Models Using Smart Card Data

Baichuan Mo\textsuperscript{a,d}, Zhenliang Ma\textsuperscript{b,*}, Haris N. Koutsopoulos\textsuperscript{c}, Jinhua Zhao\textsuperscript{d}

\textsuperscript{a}Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139
\textsuperscript{b}Department of Civil Engineering, Monash University, Melbourne, VIC 3800
\textsuperscript{c}Department of Civil and Environmental Engineering, Northeastern University, Boston, MA 02115
\textsuperscript{d}Department of Urban Studies and Planning, Massachusetts Institute of Technology, Cambridge, MA 20139

Abstract

Transit network simulation models are often used for performance and retrospective analysis of urban rail systems, taking advantage of the availability of extensive automated fare collection (AFC) and automated vehicle location (AVL) data. Important inputs to such models, in addition to origin-destination flows, include passenger path choices and train capacity. Train capacity, which has often been overlooked in the literature, is an important input that exhibits a lot of variability. The paper proposes a simulation-based optimization (SBO) framework to calibrate path choices and train capacity for urban rail systems using AFC and AVL data. The calibration is formulated as an optimization problem with a black-box objective function. Seven algorithms from four branches of SBO solving methods are evaluated. The algorithms are evaluated using an experimental design that includes five scenarios, representing different degrees of path choice randomness and crowding sensitivity. Data from the Hong Kong Mass Transit Railway (MTR) system is used as a case study. The data is used to generate synthetic observations used as “ground truth”. The results show that the response surface methods (particularly Constrained Optimization using Response Surfaces) have consistently good performance under all scenarios. The proposed approach drives large-scale simulation applications for monitoring and planning.

Keywords: Simulation-based optimization, path choice, train capacity, smart card data, urban railway systems

1. Introduction

Urban rail systems are important components of the urban transportation system. Given their high reliability and large capacity, they have attracted high passenger demand. However, high demand also leads to problems such as overcrowding and disruptions, which decrease the level of service and impact passengers. To maintain service reliability and develop efficient response strategies, it is crucial for operators to better understand passenger demand and flow patterns in the network.

Transit network loading (or simulation) models for metro systems, powered by automated collected data, provide a useful instrument for network performance monitoring. They enable operators to characterize the level of service and make decisions accordingly. A typical network loading model requires Origin-Destination (OD) matrix, supply information, and path choice fractions as input. The supply information includes the transit network topology, actual vehicle movement data, and vehicle capacity. Thanks to the wide deployment of automated fare collection (AFC) and automated vehicle location (AVL) systems, the OD demand and train movement data can be directly obtained. However, obtaining the corresponding path choices and quantifying reasonable vehicle capacity remain a challenge. According to Liu et al. (2016) and Preston et al. (2017), train capacity, defined as the maximum train load when remaining passengers in the platform denied boarding, may vary depending on the crowding levels in trains and on platforms, and passenger attitudes. The calibration of path choices and train capacity can improve the accuracy of network loading models for

*Corresponding author: Zhenliang Ma. Emails: B. Mo, baichuan@mit.edu; Z. Ma, mike.ma@monash.edu.; H. Koutsopoulos, h.koutsopoulos@northeastern.edu; J. Zhao, jinhua@mit.edu
performance monitoring. Thus, these models can provide better information to operators to adjust operating strategies, relieve congestion, and improve efficiency.

Traditionally, path choices are inferred with data from on-site surveys that are used to estimate path choice models. However, surveys are time-consuming and labor-intensive, limiting their real-world usage. To overcome these disadvantages, path choice estimation methods based on AFC data have been proposed in the literature. AFC systems provide the exact locations and times of passengers’ entry and exit transactions, which can be used to extract OD demand and passengers’ journey times. They provide rich information for analyzing passenger behavior (Koutsopoulos et al., 2019).

In an urban rail system operated near its capacity, five critical parameters are correlated with each other: OD demand, journey time, left behind (or denied boarding), path choices, and train capacity. The relationship of these parameters can be explained in Figure 1. OD demand is the input and journey time is the output (OD exit flow is a combination of the two), which can both be observed from the AFC data. Path choices, train capacity, and left behind are not observable in the AFC data. Journey time is directly affected by path choices and left behind (left behind can increase the waiting time). Left behind is directly affected by path choices and train capacity. This figure indicate the complexity of path choice estimation using AFC data. The dependencies of different parameters (e.g., path choices v.s. train capacity) should be captured.

![Figure 1: Relationship among critical parameters in urban rail systems](image)

In the context of path choice estimation, the AFC data-based methods can be categorized into two groups: path-identification methods (Kusakabe et al., 2010; Zhou and Xu, 2012; Kumar et al., 2018; Zhu et al., 2020) and parameter-inference methods (Sun and Xu, 2012; Sun et al., 2015; Zhao et al., 2017; Xu et al., 2018). The former studies aim to identify the exact path chosen by each user and even the train they boarded. Path attributes are used to evaluate how likely a path is chosen for a passenger’s trip from their observed origins to their observed destinations. The latter studies formulate probabilistic models to describe passengers’ decision-making behavior. Bayesian inference is usually used to estimate the corresponding parameters and thus derive the path choice fractions. Despite using different methods, the key components for those AFC data-based studies are similar. They all attempt to match the model-derived journey times with the observed journey times from AFC data. However, many of these studies either assume a known constant train capacity or specify a known link-impedance function. As shown in Figure 1, journey times actually depends on both path choices and train capacity. An unreasonable setting of train capacity may cause calibration bias of path choices. Simultaneous calibration of both parameters are more reasonable.

Train capacity is a vague concept. Normally trains may not reach their designed physical capacity for various reasons (e.g. passengers may decide not to board due to the crowding Liu et al. (2016)). Therefore, assuming a fixed physical capacity or fixed link-impedance function (in many previous studies) may not be a reasonable assumption in real-world situations. Only a few studies have explored the calibration of actual train capacity in the rail system. Liu et al. (2016) proposed the concept of “willingness to board” (WTB) to describe the varied capacity in a bus system, and estimated passengers’ WTB using a least square method. Xu and Yong proposed a passenger boarding model which revealed that the number of actually boarding passengers in a crowded train was closely related to the number of queuing passengers and train load. Mo et al. (2020b) proposed an effective capacity model that recognized train capacity may vary across stations depending on the corresponding number of queuing passengers and train load. The calibration of train capacity or WTB usually requires the AFC data with passengers’ boarding and journey time information. However, this information may also be affected by path choices, which were neglected in previous studies.
To fill these research gaps, we propose a simulation-based optimization (SBO) model to calibrate path choices and train capacity simultaneously and also explores the efficiency of typical SBO solution algorithms. The calibration problem is formulated as an optimization problem using AFC and AVL data. The formulation can capture the interaction among these variables and their impact on journey times. Seven optimizers (solving algorithms) from four branches of SBO solving methods are implemented for comparative analysis. They include Generic Algorithm (GA), Simulated Annealing (SA), Nelder-Mead Simplex Algorithm (NMSA), Mesh Adaptive Direct Search (MADS), Simultaneous Perturbation Stochastic Approximation (SPSA), Bayesian Optimization (BYO) and Constrained Optimization using Response Surfaces (CORS). We compare these SBO solving algorithms within a limited computational budget, defined by the number of function evaluations. Data from the Hong Kong Mass Transit Railway (MTR) system provide the foundation for a realistic case study.

The remainder of the paper is organized as follows: In Section 2, we illustrate the SBO problem formulation. Section 3 briefly describes the various SBO methods used in this study. The proposed framework is used in a case study with data from the Hong Kong MTR network in Section 4. The results are used to compare the performance of the different algorithms. Section 5 concludes the paper by summarizing the main findings and discussing future research directions.

2. Methodology

Considering the complexity of the problem and the interaction among different variables in the urban rail systems, we use a network loading model (black-box function) to capture the performance of the network for a given set of capacity and path choices. The path choice and capacity calibration model is formulated as an optimization model that attempts to minimize the error between simulated outputs (which is a function of path choices and train capacity) and the corresponding quantities observed from the AFC data.

2.1. Transit network loading model

Transit network loading (TNL) models aim to assign passengers over a transit network given the (dynamic) OD entry demand and path choices. In this study, we adopt an event-driven schedule-based TNL model proposed by Mo et al. (2020b). The model takes OD entry demand (number of tap-in passengers by time), path choices, train arrival and departure times from stations, train capacity and infrastructure information (e.g. network topology) as inputs, and outputs the passengers’ tap-out times, train loads, waiting times, and other network performance indicators of interest.

Figure 2 illustrates the main functions of the TNL model Mo et al. (2020b). Three objects are defined: train, waiting queue (on the platform), and passengers. An event is defined as a train arrival at, or departure from, a station. Events are ordered chronologically. New and transferring passengers join the waiting queue on the platform and board a train based on a first-come-first-board (FIFB) discipline. The number of successfully boarding passengers depends on the available train capacity.
The TNL model works by generating a train event list (arrivals and departures) based on the actual train movement data (AVL), and then sequentially processing the ordered events until all events are processed for the time period of interest. The processing of an individual event is based on the following rules:

- If the event is an arrival (Figure 2a), the train offloads passengers and updates its state (e.g., train load and in-vehicle passengers). Alighting passengers who need to transfer are assigned to the waiting queues on the corresponding transfer platforms (e.g., passengers transferring to platform B in Figure 2a). Passengers who tap out will be removed from the system. New tap-in passengers who entered the station between two events are added into the queue (e.g., new tap-in passengers in platform A in Figure 2a). Then, the waiting queue objects for all platforms are updated accordingly.

- If the event is a departure (Figure 2b), passengers board trains based on a FIFB priority rule. If the on-board passengers reached the train capacity, the remaining passengers at the platform will be denied boarding. Finally, the state of the train (train load and in-vehicle passengers) and the waiting queue at the platform are updated accordingly.

2.2. Problem formulation

Consider a general urban rail network in a specific time period $T$, represented as $G = (S, A)$, where $S$ is the set of stations and $A$ is the set of directed links. We divide $T$ into several time intervals with equal
length $\tau$ (e.g., $\tau = 15$ min). Denote the set of all time intervals as $T = \{1, 2, ..., T/\tau\}$. Define a time-space (TS) node as $i_m$, where $i \in S$ and $m \in T$. $i_m$ represents station $i$ in time interval $m$.

For an OD pair $(i, j)$ ($i, j \in S$), the OD entry flow ($q_{i,m,j}^e$) represents the number of passengers entering station $i$ during time interval $m$ and exiting at station $j$. Let the set of all OD entry flows be $q^e$. The OD exit flow ($q_{i,m,j}^x$) represents the number of passengers who exit at station $j$ in the time interval $m$ with origin $i$. $q_{i,m,j}^e$ and $q_{i,m,j}^x$ are inputs and outputs of the TNL model, respectively.

Let the set of all paths between $(i, j)$ be $R(i, j)$. We assume that the path choice behavior can be formulated as a C-logit model (Cascetta et al., 1996), which is an extension of the multinomial logit (MNL) model to correct the correlation among paths due to overlapping (Prato, 2009). The path choice fraction for path $r \in R(i, j)$ in time interval $m$ ($p_{i,m,j}^r$) is formulated as:

$$p_{i,m,j}^r = \frac{e^{\beta r \cdot (X_{r,m} + \beta CF \cdot CF_r)}}{\sum_{r' \in R(i, j)} e^{\beta r' \cdot (X_{r',m} + \beta CF \cdot CF_{r'})}}, \quad \forall r \in R(i, j), m \in T, i, j \in S$$

(1)

where $\mu$ is the scale parameter of the Gumbel distribution of the error term (Ben-Akiva et al., 1985), which is usually normalized to 1. Larger (smaller) $\mu$ means the choice behavior is more deterministic (random). $X_{r,m}$ is the vector of attributes for path $r$ in time interval $m$ (e.g., in-vehicle time, number of transfers, transfer walking time, etc.). $CF_r$ is the commonality factor of path $r$ which measures the degree of similarity of path $r$ with the other paths of the same OD. $\beta_X$ and $\beta_{CF}$ are the corresponding coefficients to be estimated. Let $\beta$ be the vector that combining $\beta_X$ and $\beta_{CF}$ (i.e. $\beta = [\beta_X, \beta_{CF}]$).

$CF_r$ is defined as:

$$CF_r = \ln \sum_{r' \in R(i, j)} \left( \frac{D_{r,r'}}{D_r D_{r'}} \right)^{\gamma}$$

(2)

where $D_{r,r'}$ is the number of common stations of path $r$ and $r'$. $D_r$ and $D_{r'}$ are the number of stations for path $r$ and $r'$, respectively. $\gamma$ is a fixed positive parameter. Let the set of all path choice fractions be $p$.

The values of $\beta$ can be bounded from above and below. The boundaries can be obtained from the prior knowledge and previous survey results. Denote the upper bound as $U_\beta$ and lower bound as $L_\beta$ ($L_\beta \leq \beta \leq U_\beta$), where $U_\beta$ and $L_\beta$ are both vectors with the same cardinality as $\beta$.

According to Mo et al. (2020b), the actual train capacity utilized by passengers is determined by three factors: a) waiting passenger distribution on the platform, b) train load and distribution across the train, and c) passengers’ willingness to board a crowded train. Thus, train capacity is not constant. Instead, it is dynamic and changes across stations and trains depending on the crowding level of the train and the platform. Mo et al. (2020b) model the capacity of train $k$ at station $i$ ($C_{k,i}$) as:

$$C_{k,i} = \begin{cases} \theta_0 n_i + \theta_1 H_{k,i} + \theta_2 Q_{k,i} & \text{if station } i \text{ is in the list of congested stations} \\ \theta_0 n_i & \text{otherwise} \end{cases} \quad \forall k, i$$

(3)

where $n_i$ is the number of cars of train $i$. $H_{k,i}$ is the load of train $k$ when it arrives at station $i$. $Q_{k,i}$ is the number of passengers waiting on the platform when train $k$ arrives at station $i$. $\theta_0$, $\theta_1$, and $\theta_2$ are parameters to be estimated ($\theta_0, \theta_1, \theta_2 > 0$). $\theta_0 n_i$ is the base capacity, which can be seen as the train load that represents acceptable service standards. At uncongested stations, passengers are assumed not to board when the train load is greater than $\theta_0 n_i$. At congested stations, passengers may still board a train even if it is already crowded (Liu et al., 2016), which makes the effective train capacity higher than $\theta_0 n_i$.

In the discussion that follows, let $\theta$ be the vector of these three parameters. We assume that the values that these parameters can take is $L_\theta \leq \theta \leq U_\theta$, where $L_\theta$ and $U_\theta$ are the corresponding lower and upper bounds, respectively.

The goal is to calibrate $\theta$ and $\beta$ vectors (used by the TNL model) based on indirect observations. Two sets of observations are used for the calibration: observed OD exit flows and observed journey time distribution (JTD). Both of them can be obtained from the AFC data.

Let the ground truth (observed) OD exit flow be $q_{i,m,j}^x$. Let $\hat{f}_{i,j,t}(x)$ be the model-derived JTD of passengers with origin $i$ who exit at station $j$ during time interval $t$. Let $\hat{f}_{i,j,t}(x)$ be the corresponding observed JTD extracted from the AFC data. Since $f_{i,j,t}(x)$ and $\hat{f}_{i,j,t}(x)$ are estimated from passengers’ journey time
observations, only the OD pairs with more than $E$ passengers exiting in a specific time interval are considered, where $E$ is a predetermined threshold to ensure enough sample size. Denote the set of corresponding OD pairs and exit time intervals as $E$, where $E = \{ (i, j_n) : q^{i, j_n} \neq 0, \forall i, j \in S, n \in T \}$.

The calibration problem is formulated as an optimization problem:

$$
\begin{align*}
\min_{\beta, \theta} \quad & w_1 \sum_{i,j \in S, m \in T} (q^{i,j} - \tilde{q}^{i,j})^2 + w_2 \sum_{(i,j,n) \in E} D_{\text{KL}}(f_{i,j,n} \| \tilde{f}_{i,j,n}) \\
\text{s.t.} \quad & q^{i,j} = \text{TNL}(p, q^e, \theta) \quad \forall i, j \in S, m \in T, \\
& \tilde{f}_{i,j,n}(x) = \text{TNL}(p, q^e, \theta) \quad \forall (i, j_n) \in E, \\
& p^{r,m,j}_{r'} = \frac{e^{\mu(x-m+\beta CF_{CF})}}{\sum_{r' \in R(i,j)} e^{\mu(x-m+\beta CF_{CF})}} \quad \forall p^{r,m,j}_{r'} \in p, \\
& L_\beta \leq \beta \leq U_\beta, \\
& L_\theta \leq \theta \leq U_\theta 
\end{align*}
$$

The objective function (Eq. 4a) has two parts: the square error between model-derived OD exit flows and the corresponding observations, and the difference between model-derived and observed JTD. $w_1$ and $w_2$ are weights used to balance the scale and the importance of the two parts. The difference of the two distributions is expressed using Kullback-Leibler (KL) divergence ($D_{\text{KL}}$):

$$
D_{\text{KL}}(f_{i,j,n} \| \tilde{f}_{i,j,n}) = \int f_{i,j,n}(x) \cdot \log \frac{f_{i,j,n}(x)}{\tilde{f}_{i,j,n}(x)} dx. 
$$

TNL($p, q^e, \theta$) is the black-box function that corresponds to the TNL model, which can output the model-derived OD exit flows and JTD for a given set of path choices and train capacity. Since the TNL model has no analytic form, Eq. 4 is a SBO problem with upper and lower bound constraints. In the following section, we discuss seven different algorithms appropriate for the solution of SBO problems. These algorithms belong to four general approach of SBO solving methods.

### 3. Simulation-based optimization algorithms

There are four major classes of methods for solving the SBO problems, including the heuristic methods, direct search methods, gradient-based methods, and response surface methods (Osorio and Bierlaire, 2013; Amaran et al., 2016). Heuristic methods are partial search algorithms that may provide a sufficiently good solution to an optimization problem, especially with incomplete or imperfect information or limited computation capacity. Direct search methods are derivative-free methods that are based on the sequential examination of trial points generated by a certain strategy. They are attractive as they are easy to describe and implement. More importantly, they are suitable for objective functions where gradients do not exist everywhere. Gradient-based approaches (or stochastic approximation methods) attempt to optimize the objective function using estimated gradient information. These methods aim to imitate the steepest descent methods in derivative-based optimization. Finite difference schemes can be used to estimate gradients but they may involve a large number of expensive function evaluations if the number of decision variables is large. Response surface methods are useful in the context of continuous optimization problems. They focus on learning input-output relationships to approximate the underlying simulation by a pre-defined functional form (also known as a meta-model or surrogate model). This functional form can then be used for optimization leveraging powerful derivative-based optimization techniques.

In this study, we use seven representative algorithms belonging to these four classes of SBO methods to address the aforementioned path choice and train capacity calibration problem. Table 1 summarizes the main characteristic of these algorithms. The summary of all algorithms is described in Table 1.

In the discussion that follows, let $\Theta$ be the combined vector of $\beta$ and $\theta$ (i.e., $\Theta = [\beta, \theta]$ is the vector of all coefficients to be estimated). Let $N$ be the dimension of $\Theta$ (i.e. $\Theta \in \mathbb{R}^N$).
### Table 1: Algorithms Summary

| Type                | Algorithm                                      | Constraints | Stochastic | Source                     |
|---------------------|------------------------------------------------|-------------|------------|----------------------------|
| Heuristic method    | Genetic Algorithm (GA)                         | Yes         | Yes        | Fortin et al. (2012)       |
|                     | Simulated Annealing (SA)                       | Yes         | Yes        | Tsallis and Stariolo (1996) |
| Direct search       | Nelder-Mead Simplex Algorithm (NMSA)           | No          | No         | Gao and Han (2012)         |
|                     | Mesh Adaptive Direct Search (MADS)             | Yes         | Yes        | Abramson et al. (2009)     |
| Gradient-based      | Simultaneous Perturbation                      | Yes         | Yes        | Spall et al. (1992)        |
| Response surface    | Bayesian Optimization (BYO)                    | Yes         | Yes        | Snoek et al. (2012)        |
|                     | Constrained Optimization using Response Surfaces (CORS) | Yes | Yes | Regis and Shoemaker (2005) |

3.1. Genetic Algorithm (GA)

GA is a heuristic method for solving both constrained and unconstrained optimization problems, which belongs to the larger class of evolutionary algorithms inspired by natural selection, the process that drives biological evolution. The GA repeatedly modifies a population of individual solutions as an evolution process (Whitley, 1994). The GA can be used to solve a variety of optimization problems that are not well suited for standard optimization algorithms, such as the SBO problem where the objective function (or constraints) is nondifferentiable and highly nonlinear.

The evolution starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. In each generation, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population “evolves” toward an optimal solution. The genetic algorithm uses three main procedures at each step to create the next generation from the current population: 1) Selection: select the individuals, called parents, that contribute to the population at the next generation. Individuals with better objective function values are more likely to be selected. 2) Crossover: combine two parents to form children for the next generation. 3) Mutation: apply random changes to individual parents to form children.

In this study, we adopted a blend crossover and Gaussian mutation methods. The probability of crossover is set as 0.8 and the probability of mutating is set as 0.4. And the population size is set as 6 given the limited computational budget. The algorithm is implemented by the Python `deap` package (Fortin et al., 2012).

3.2. Simulated Annealing (SA)

SA is a heuristic method for solving optimization problems (Van Laarhoven and Aarts, 1987). The method is based on the physical process of heating a material and then slowly lowering the temperature to decrease defects, thus minimizing the system energy.

At each iteration of the SA algorithm, a new point is randomly generated. The distance of the new point from the current point, or the extent of the search, is based on a probability distribution with a scale proportional to the temperature. A distorted Cauchy-Lorentz visiting distribution is used in this study (Tsallis and Stariolo, 1996). The algorithm accepts all new points that lower the objective function, but also, with a certain probability, points that raise the objective function. By accepting points that raise the objective function, the algorithm avoids being trapped in local minima. An annealing schedule is selected to systematically decrease the temperature as the algorithm proceeds. As the temperature decreases, the algorithm reduces the extent of its search to converge to a minimum.

In this study, the SA algorithm in Python `scipy` package is adopted for the implementation with all model parameters set as default (SciPy, 2019).

3.3. Nelder-Mead Simplex Algorithm (NMSA)

NMSA is a simplex method for finding a local minimum (Nelder and Mead, 1965). NMSA in N dimensions maintains a set of N + 1 test points arranged as a simplex. Denote the initial value of Θ as Θ\textsuperscript{ini}. The initial simplex set (N + 1 points) is generated as \{Θ : Θ = Θ\textsuperscript{ini} + e\textsubscript{i}, i = 1, ..., N\} ∪ {Θ\textsuperscript{ini}}, where e\textsubscript{i} ∈ R\textsuperscript{N} is the unit vector in the i\textsuperscript{th} coordinate, σ is the step-size which is set as 0.05 in this study (Gao and Han, 2012).
Based on the initial simplex, the model evaluates the objective function for each test point, in order to find a new test point to replace one of the old test points. The new candidate can be generated through simplex centroid reflections, contractions, or other means depending on the function value of the test points. The process will generate a sequence of simplexes, for which the function values at the vertices get smaller and smaller. The size of the simplex is reduced and finally, the coordinates of the minimum point are found.

Four possible operations: reflection, expansion, contraction, and shrink are associated with the corresponding scalar parameters: $\alpha_1$ (reflection), $\alpha_2$ (expansion), $\alpha_3$ (contraction) and $\alpha_4$ (shrink). In this study, we set the value of these parameters as $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\} = \{1, 2, 0.5, 0.5\}$ as suggested in Gao and Han (2012). The algorithm is implemented by the Python scikit-learn package with all parameters set as default. Since NMSA is designed for unconstrained problem, we turned the bound of $\Theta$ into a big penalized term in the objective function for this algorithm. More details regarding the NMSA can be referred to (Gao and Han, 2012).

### 3.4. Mesh Adaptive Direct Search (MADS)

The MADS algorithm is a directional direct search framework for nonlinear optimization (Audet and Dennis Jr, 2006). It seeks to improve the current solution by testing points in the neighborhood of the current point (the incumbent). The neighborhood points are generated by moving one step in each direction from the incumbent on an iteration-dependent mesh. Each iteration of MADS consists of a SEARCH stage and an optional POLL stage. The SEARCH stage evaluates a finite number of points proposed by the searching strategy (e.g. moving one step around from the current point). Whenever the SEARCH step fails to generate an improved mesh point, the POLL step is invoked. The POLL step conducts local exploration near the current incumbent, which also intends to find an improved point on the mesh. Once an improved point is found, the algorithm updates the current point and constructs a new mesh. According to (Audet and Dennis Jr, 2006), the mesh size parameters approach zero as the number of iteration approaches to infinity, which demonstrates the convergence of the MADS algorithm.

In this paper, we use a variant of the MADS method called ORTHO-MADS, which leverages a special orthogonal positive spanning set of polling directions. More details regarding the algorithm can be found in (Abramson et al., 2009). NOMAD 3.9.1 (Audet et al., 2009) with the Python interface is used for the MADS algorithm application. The hyper-parameters are tuned based on the NOMAD user guide. The direction type is set as orthogonal, with $N + 1$ directions generated at each poll. Latin Hypercube search is not applied.

### 3.5. Simultaneous Perturbation Stochastic Approximation (SPSA)

SPSA is a descent direction method for finding local minimum. It approximates the gradient with only two measurements of the objective function, regardless of the dimension of the optimization problem. Denote the objective function in Eq. 4 as $Z(\Theta)$. The estimated parameters in the $k$-th iteration is denoted as $\Theta^{(k)}$. Then one iteration for the SPSA is performed as

$$
\Theta^{(k+1)} = \Theta^{(k)} - a_k \cdot \hat{\nabla}Z(\Theta^{(k)})
$$

where

$$
\hat{\nabla}Z(\Theta^{(k)}) = \frac{Z(\Theta^{(k)} + c_k \Delta_k) - Z(\Theta^{(k)} - c_k \Delta_k)}{2c_k \Delta_k}
$$

$$
a_k = \frac{a}{(k + 1 + A)\alpha}
$$

$$
c_k = \frac{c}{(k + 1)\gamma}
$$

$\Delta_k$ is a random perturbation vector, whose elements are obtained from a Bernoulli distribution with the probability parameter equal to 0.5. $\{\alpha, \gamma, a, c, A\}$ are tuned as $\{0.602, 0.101, 0.001, 0.007, 0.1M\}$ in this study according to the numerical tests and guidelines from prior empirical studies (Gomez-Dans, 2012). $M$ is the maximum number of iterations.
3.6. Bayesian Optimization (BYO)

BYO constructs a probabilistic model of the objective function and exploits this model to determine where to evaluate the objective function for the next step. The philosophy of BYO is to use all of the information available from previous evaluations, instead of simply relying on the local gradient and Hessian approximations. This enables BYO to find the minimum of difficult non-convex functions with relatively few function evaluations.

BYO assumes a prior distribution for the objective function values and uses an acquisition function to determine the next point to evaluate. In this study, we use the Gaussian process as the prior distribution for the objective function due to its flexibility and tractability. For the acquisition function, we tested three common criteria: probability of improvement (POI), expected improvement (EI), and upper confidence bound (UCB) (Snoek et al., 2012). The EI criterion is used in this path choice calibration problem due to its best performance in our problem. The BYO is implemented in Python with bayes_opt package. More details regarding the BYO can be found in (Snoek et al., 2012).

3.7. Constrained Optimization using Response Surfaces (CORS)

CORS is a response surface method for global optimization. In each iteration, it updates the response surface model based on all previously probed points and selects the next point to evaluate. The principles for next point selection are (a) finding new points that have lower objective function value, and (b) improving the fitting of the response surface model by sampling feasible regions where little information exists. Hence, the next point is selected by solving the minimization problem of the current response surface function subject to constraints that the next point should be more than a certain distance away from all previous points (Regis and Shoemaker, 2005).

An algorithm following the CORS framework requires two components: (a) a scheme for selecting an initial set of points for objective function evaluation and (b) a procedure for globally approximating the objective function (i.e. a response surface model). In this study, the initial sampling is conducted using the Latin hypercube methods, with the initial sampling number equal to $0.2 \times$ the total number of function evaluations allowed. The radial basis function (RBS) is used as the response surface model. For the subsequent sampling, a modified version of the CORS algorithm with space re-scaling is used. Details about the algorithm can be found in Regis and Shoemaker (2005) and Knysh and Korkolis (2016).

4. Case study

The proposed modeling framework is tested using data from the Hong Kong MTR network.

4.1. Hong Kong MTR Network

MTR is the operator of the Hong Kong urban rail network, which provides services for the urbanized areas of Hong Kong Island, Kowloon, and the New Territories. The system currently consists of 11 lines with 218.2 km (135.6 miles) of rail, serving 159 stations including 91 heavy rail stations and 68 light rail stops. It serves over 5 million trips on an average weekday. Most of the passengers use a smart card fare-payment system named Octopus. For the urban heavy rail lines, trip transactions are recorded when passengers enter and exit the system, providing information about the tap-in and tap-out stations and corresponding timestamps.

4.2. Experimental design

We use AFC data on a typical weekday afternoon peak period (18:00-19:00) in March 2017 for the model application. Li (2014) conducted a revealed-preference (RP) path choice survey of more than 20,000 passengers in the MTR system and used them to estimate a path choice model. The estimation results are shown in Appendix A. The following attributes were used in the specification of the model: (a) total in-vehicle time, (b) the number of transfer times, (c) relative walking time (total walking time divided by total path distance) and (d) the commonality factor (Eq. 2).

As the real-world path choice information and train capacity are usually unavailable, we validate the models with synthetic data. To generate the synthetic data, we first extract the OD entry flow $(q_{i,m}^{o,j})$ from the real-world AFC records. We assume a synthetic $\Theta$ as the “true” path choice and train capacity.
parameters. The TNL model with the true OD entry flow, train timetable, and the synthetic T as inputs is used to simulate the travel of passengers in the system and record people’s tap-in and tap-out time. The input timetable is treated as the synthetic AVL data. The resulting passengers’ tap-in and tap-out times are treated as the synthetic AFC data. The synthetic data, including “true” passenger path choices and train capacity, are used to evaluate the performance of the model under the various solution algorithms.

To compare the different SBO solving algorithms, we design five test scenarios summarized in Table 2. Each scenario has a different synthetic Θ. The selection of synthetic Θ can represents different assumptions about passengers’ choice behavior and sensitivity to crowding. For the reference scenario, we use the path choice parameters in Table A.8 as the synthetic β, and use the estimated train capacity parameters in Mo et al. (2020b) as the synthetic θ.

### Table 2: Scenario design

| Parameter category | Synthetic Θ | Scenarios | Bound  |
|--------------------|-------------|-----------|--------|
|                    |             | Reference | Path choice | Train capacity |        |
|                    |             |           | Random | Deterministic | Crowding-sensitive | Crowding-insensitive |
| In vehicle time    | -0.147      | 0         | -2.0   | -0.147        | -0.147            | [-2, 0]            |
| Relative walking time | -1.271    | 0         | -5.0   | -1.271        | -1.271            | [-5, 0]            |
| Number of transfers | -0.573     | 0         | -3.0   | -0.573        | -0.573            | [-3, 0]            |
| Commonality factor | -3.679      | 0         | -10.0  | -3.679        | -3.679            | [-10, 0]           |
| θ₀                | 232         | 232       | 232    | 225           | 235               | [220, 260]         |
| θ₁                | 0.0732      | 0.0732    | 0.0732 | 0.2           | 0                 | [0, 0.2]           |
| θ₂                | 0.0607      | 0.0607    | 0.0607 | 0.2           | 0                 | [0, 0.2]           |

Passengers’ actual path choice behavior are assumed to be random (each path is equally likely to be selected) or deterministic. For the random path choice scenario, we set all synthetic choice parameters as 0, which means all available paths are equally likely to be chosen. For the deterministic path choice scenario, we set all synthetic choice parameters as the lower bounds (i.e. the maximum absolute value possible). Under this scenario, a slight difference in attributes between two paths can lead to a high difference in choice probability (i.e. this is close to passengers following the shortest path). As for the train capacity, the synthetic θ for these two scenarios is the same as the reference scenario.

Passengers’ sensitivity to crowding may also vary. If all passengers are not sensitive to the crowding, train capacity can be modeled as a fixed value. However, if passengers become more sensitive to the crowding, the actual train capacity may largely depend on the crowding level in the train and on the platform. Therefore, passengers’ sensitivity to crowding can be reflected by the scale of θ₁ and θ₂ (Mo et al., 2020b). For the crowding-sensitive scenario, we set the synthetic train capacity parameters as θ₀ = 225, θ₁ = 0.2, θ₂ = 0.2. Compared to the reference scenario, θ₁ and θ₂ are higher to represent higher sensitivity. And θ₀ is decreased to offset the capacity increasing caused by the increase of θ₁ and θ₂. As for the crowding-insensitive scenario, we set the synthetic train capacity parameters as θ₀ = 235, θ₁ = 0, θ₂ = 0, which can be seen as a fixed-capacity model.

### 4.3. Case study settings

The lower and upper bounds of all parameters (Lβ, Uβ, Lθ, Uθ) are shown in Table 2. Θ is set as (Lθ + Uθ)/2 for all scenarios. To compare different algorithms, a fixed computational budget, 100 function evaluations, is applied to all algorithms. All algorithms except for NMSA (deterministic algorithm) are replicated for 5 times (with different random seeds) to decrease the impact of randomness.

### 4.4. Reference scenario results

The convergence results of the reference scenario are depicted in Figure 3. Each point represents the average value over all replications. We found that the performance of different algorithms varied. Given

---

1The word “deterministic” here just represents the degree of randomness is low. The “truly” deterministic corresponds to all parameters go to $-\infty$. 

---

10
the limited number of function evaluations, CORS, BYO, and SPSA converge to a relatively small objective function. GA, MADS, and SA have relatively large objective functions values upon termination. In terms of convergence speed, the response surface methods (BYO and CORS) have the fastest convergence speed. They also reach the lowest objective function value. This is consistent with conclusions regarding the performance of the SBO algorithms when used in the transportation domains (Osorio and Bierlaire, 2013; Cheng et al., 2019; Mo et al., 2020a).

Figure 3 also summarizes the behavior of the algorithm stability. The vertical line indicates the \( \frac{1}{4} \times \) standard deviations over the five replications. NMSA is a deterministic algorithm and not affected by randomness. BYO and CORS show high randomness in the first half iterations. However, as the number of function evaluations increases, the standard deviation of the objective function decreases, and the results become stable. GA, SA, and MADS are unstable compared to other algorithms. This means that the heuristic algorithms (GA and SA) are not suitable for the calibration problem studied in this paper. The instability of MADS may be because it may converge to non-stationary points (Abramson and Audet, 2006).

Table 3 compares the parameters estimated by different algorithms with the synthetic ones. Although some algorithms can reach similar objective function values, they result in different estimated parameters. For example, CORS and SPSA have similar objective function values. However, SPSA performs better in path choice estimation while CORS performs better in train capacity estimation. We also observe that the train capacity parameters are relatively harder to estimate. This may be because most of the stations in the rail system are not congested and all passengers can board the trains. Thus, the objective function is not very sensitive to the train capacity parameters.
Table 3: estimation results of the reference scenario

| Category          | Variable name | "True" | Estimated parameters |
|-------------------|---------------|--------|----------------------|
|                   |               | GA     | SA                   | NMSA | MADS | SPSA | BYO   | CORS  |
| Path choice       | In-vehicle time | -0.147 | -0.392              | -0.327 | -0.342 | -0.454 | -0.170 | -0.207 | -0.229 |
|                   | Relative walking time | -1.271 | -2.205              | -3.010 | -3.020 | -0.302 | -2.257 | -2.493 | -2.486 |
|                   | Number of transfers | -0.573 | -1.143              | -0.787 | -0.389 | -1.248 | -0.598 | -0.176 | -0.756 |
|                   | Commonality factor | -3.679 | -6.482              | -6.851 | -7.250 | -7.834 | -4.419 | -5.434 | -5.716 |
| Train capacity    | $\theta_0$    | 232    | 239                | 243    | 259    | 252    | 241    | 234    | 243    |
|                   | $\theta_1$    | 0.073  | 0.117              | 0.118  | 0.146  | 0.040  | 0.162  | 0.110  | 0.069  |
|                   | $\theta_2$    | 0.061  | 0.069              | 0.110  | 0.080  | 0.080  | 0.163  | 0.100  | 0.086  |
| Objective function|               | -      | 676,392            | 416,923 | 359,663 | 773,526 | 245,269 | 258,688 | 203,885 |

4.5. Sensitivity analysis

4.5.1. Impact of randomness in path choice behavior

Figure 4 shows the estimation results for two path choice related scenarios: random and deterministic. The estimated parameters are shown in Table 4 and 5. For the random scenario, all "true" (synthetic) path choice parameters are set as zero, which means all paths are equally likely to be chosen. We observe that, in this scenario (Figure 4a), CORS and SA algorithms perform the best with the lowest objective function. Compared to the reference scenario in Section 4.4, the decreased performance of BYO and SPSA may be because to the "true" $\beta$ is close to the upper-bound ($U_\beta = 0$). The Gaussian posterior distribution in BYO and gradient estimation in SPSA can suffer from instability in the boundary. From Table 4, we observe the parameters of in-vehicle time and number of transfers are better estimated than those of relative walking time and commonality factors.

Figure 4b shows the results of the deterministic scenario. The initial objective function is relatively small ($1.5 \times 10^5$) compared to the reference scenario ($1.5 \times 10^6$). All algorithms only reduce the objective function by around $\frac{1}{3}$ except for the CORS algorithm. The good performance of CORS may come from the global searching with the Latin hypercube method. It is better suited to explore the points near boundaries. Although the objective function does not decrease too much, the estimated parameters are still acceptable (see Table 5).

Table 4: estimation results of the random path choice scenario

| Category          | Variable Name | "True" | Estimated Parameters |
|-------------------|---------------|--------|----------------------|
|                   |               | GA     | SA                   | NMSA | MADS | SPSA | BYO   | CORS  |
| Path choice       | In-vehicle time | 0      | 0                   | -0.072 | -0.050 | 0 | -0.108 | -0.037 | 0    |
|                   | Relative walking time | 0      | -2.151              | -1.139 | -1.807 | -1.000 | -1.719 | -3.725 | -0.702 |
|                   | Number of transfers | 0      | -0.348              | -0.185 | -0.435 | -1.334 | -0.631 | -0.207 | 0    |
|                   | Commonality factor | 0      | -5.997              | -1.945 | -9.991 | -5.432 | -5.127 | -4.155 | -8.000 |
| Train capacity    | $\theta_0$    | 232    | 243                | 224    | 254    | 232    | 241    | 248    | 223    |
|                   | $\theta_1$    | 0.073  | 0.067              | 0.050  | 0.124  | 0.048  | 0.106  | 0.079  | 0.016  |
|                   | $\theta_2$    | 0.061  | 0.037              | 0.072  | 0.136  | 0.134  | 0.112  | 0.159  | 0.072  |
| Objective function|               | -      | 1,202,761          | 756,321 | 1,399,836 | 1,365,291 | 1,429,942 | 1,203,696 | 855,627 |

Table 5: estimation results of the deterministic path choice scenario

| Category          | Variable Name | "True" | Estimated Parameters |
|-------------------|---------------|--------|----------------------|
|                   |               | GA     | SA                   | NMSA | MADS | SPSA | BYO   | CORS  |
| Path choice       | In-vehicle time | -2     | -1.240              | -1.243 | -1.205 | -1.160 | -1.544 | -1.537 | -1.830 |
|                   | Relative walking time | -5     | -3.180              | -3.358 | -2.819 | -2.480 | -3.728 | -3.807 | -4.492 |
|                   | Number of transfers | -3     | -1.575              | -1.551 | -1.419 | -1.524 | -1.786 | -1.761 | -2.661 |
|                   | Commonality factor | -10    | -5.307              | -5.251 | -4.735 | -4.920 | -6.346 | -6.379 | -8.819 |
| Train capacity    | $\theta_0$    | 232    | 237                | 232    | 228    | 237    | 239    | 232    | 237    |
|                   | $\theta_1$    | 0.073  | 0.095              | 0.076  | 0.095  | 0.180  | 0.097  | 0.110  | 0.123  |
|                   | $\theta_2$    | 0.061  | 0.101              | 0.069  | 0.091  | 0.062  | 0.110  | 0.106  | 0.093  |
| Objective function|               | -      | 125,100            | 128,157 | 118,915 | 135,922 | 113,805 | 124,448 | 63,220 |
4.5.2. Impact of crowding sensitivity

Figure 5 shows the estimation results of the two scenarios related to train capacity (i.e., crowding-sensitive and crowding-insensitive). In the crowding-insensitive scenario (Figure 5a), the conclusions are similar to the reference scenario. CORS, BYO, NMSA, and SPSA converge to low objective function values and outperform other algorithms. The performance of NMSA and MADS is improved compared to the reference scenario. In the crowding-sensitive scenario, we still observe a good performance by the CORS, NMSA, and SPSA algorithms. The performance of BYO is slightly reduced. The results shown in Table 6 and 7 indicate that $\theta_0$ (base capacity) is hard to estimate. This may be because trains at most stations do not reach the capacity. Therefore, for many OD pairs, the OD exit flows (directly related to the objective function) are not sensitive to the base capacity parameter.
Figure 5: Algorithm performance in the two train capacity scenarios

Table 6: estimation results of the crowding-insensitive train capacity scenario

| Category     | Variable Name       | "True" | GA    | SA    | NMSA  | MADS  | SPSA  | BYO   | CORS  |
|--------------|---------------------|--------|-------|-------|-------|-------|-------|-------|-------|
| Path choice  | In-vehicle time     | -0.147 | -0.392| -0.181| -0.254| -0.460| -0.191| -0.197| -0.177|
|              | Relative walking    | -1.271 | -2.153| -2.044| -2.636| -2.294| -2.284| -2.469| -2.025|
|              | Number of transfers | -0.573 | -1.127| -1.614| -1.279| -0.490| -0.760| -0.908| -1.011|
|              | Commonality factor  | -3.679 | -6.489| -6.500| -7.492| -7.750| -5.299| -5.474| -5.130|
| Train capacity | $\theta_0$         | 235    | 239   | 245   | 249   | 230   | 241   | 238   | 236   |
|              | $\theta_1$         | 0      | 0.088 | 0.109 | 0.096 | 0.050 | 0.096 | 0.093 | 0.084 |
|              | $\theta_2$         | 0      | 0.05  | 0.058 | 0.108 | 0.050 | 0.150 | 0.078 | 0.063 |
| Objective function |              | -  | 700,441| 418,196| 277,835| 553,765| 241,533| 305,846| 277,212|

14
Table 7: estimation results of the crowding-sensitive train capacity scenario

| Category     | Variable Name        | "True"    | Estimated Parameters |
|--------------|----------------------|-----------|----------------------|
| Path choice  | In-vehicle time      | -0.147    | -0.472 -0.217 -0.228 -0.332 -0.177 -0.195 -0.196 |
|              | Relative walking time | -1.271    | -2.533 -1.575 -2.735 -1.568 -2.118 -1.763 -2.534 |
|              | Number of transfers  | -0.573    | -0.759 -1.169 -1.323 -0.816 -0.495 -0.892 -0.734 |
|              | Commonality factor   | -3.679    | -6.489 -6.324 -7.040 -7.834 -4.361 -6.046 -5.021 |
| Train capacity | $\theta_0$          | 225       | 238 244 238 245 244 237 237 |
|              | $\theta_1$          | 0.2       | 0.166 0.149 0.121 0.112 0.140 0.085 0.099 |
|              | $\theta_2$          | 0.2       | 0.080 0.114 0.125 0.144 0.129 0.123 0.110 |
| Objective function | -    | 765,621   | 320,745 231,228 502,341 199,753 335,558 169,057 |

5. Conclusion

In this paper, we propose an SBO framework to calibrate train capacity and path choice model parameters simultaneously for metro systems using AFC and AVL data. The advantage of the proposed framework lies in capturing the collective effect of both path choices and train capacity on passenger journey times. Seven representative algorithms from four main branches of SBO methods are applied and compared with respect to their solution accuracy, convergence speed, and stability. We applied the proposed framework using data from the Hong Kong MTR network and compared the performance of the different algorithms. Overall, the results show that some algorithms result in a reasonable estimation of the parameters of interest. These results also support the effectiveness of the proposed SBO framework for calibrating these key parameters using AFC and AVL data. Especially, the response surface methods (particularly CORS) exhibit consistently good performance. The SBO framework is flexible to accommodate a wide range of path choice and train capacity models.

This paper has some limitations. First, we validate the framework and evaluate the algorithmic performance only using synthetic AFC and AVL data. Therefore, the complexities of noise and uncertainties in actual data do not play any role. This is caused by the absence of real-world path choice and train capacity information. Future research can collect real-world path choice and train capacity data to conduct more realistic model validation. Second, we assumed that the path choice behavior is similar for the whole network (same $\beta$ values). Given the real-world path choice behavior is possibly more diverse and heterogeneous, future research can explore clustering different OD pairs with different $\beta$ values.

6. Author contribution statement

The authors confirm contribution to the paper as follows: study conception and design: B. Mo, Z. Ma, H.N. Koutsopoulos, J. Zhao; data collection: B. Mo, Z. Ma; analysis and interpretation of results: B. Mo, Z. Ma, H.N. Koutsopoulos; draft manuscript preparation: B. Mo, H.N. Koutsopoulos. All authors reviewed the results and approved the final version of the manuscript.

7. Acknowledgements

The authors would like to thank the Hong Kong Mass Transit Railway (MTR) for their support and data availability for this research.

References

Abramson, M.A., Audet, C., 2006. Convergence of mesh adaptive direct search to second-order stationary points. SIAM Journal on Optimization 17, 606–619.
Abramson, M.A., Audet, C., Dennis Jr, J.E., Digabel, S.L., 2009. Orthomads: A deterministic mads instance with orthogonal directions. SIAM Journal on Optimization 20, 948–966.
Amaran, S., Sahinidis, N.V., Sharda, B., Bury, S.J., 2016. Simulation optimization: a review of algorithms and applications. Annals of Operations Research 240, 351–380.
Audet, C., Dennis Jr, J.E., 2006. Mesh adaptive direct search algorithms for constrained optimization. SIAM Journal on optimization 17, 188–217.
Audet, C., Le Digabel, S., Tribes, C., 2009. Nomad user guide. Rapport technique.
Ben-Akiva, M.E., Lerman, S.R., Lerman, S.R., 1985. Discrete choice analysis: theory and application to travel demand. volume 9. MIT press.
Cascetta, E., Nuzzolo, A., Russo, F., Vitetta, A., 1996. A modified logit route choice model overcoming path overlapping problems. specification and some calibration results for interurban networks, in: Proceedings of The 13th International Symposium On Transportation And Traffic Theory.
Cheng, Q., Wang, S., Liu, Z., Yuan, Y., 2019. Surrogate-based simulation optimization approach for day-to-day dynamics model calibration with real data. Transportation Research Part C: Emerging Technologies 105, 422–438.
Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagné, C., 2012. DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research 13, 2171–2175.
Gao, F., Han, L., 2012. Implementing the nelder-mead simplex algorithm with adaptive parameters. Computational Optimization and Applications 51, 259–277.
Gomez-Dans, J., 2012. A simultaneous perturbation stochastic approximation optimisation code in python.
Knysh, P., Korkolis, Y., 2016. Blackbox: A procedure for parallel optimization of expensive black-box functions. arXiv preprint arXiv:1605.00998.
Koutsopoulos, H.N., Ma, Z., Noursalehi, P., Zhu, Y., 2019. Chapter 10 - transit data analytics for planning, monitoring, control, and information, in: Antoniou, C., Dimitriou, L., Pereira, F. (Eds.), Mobility Patterns, Big Data and Transport Analytics. Elsevier, pp. 229 – 261. URL: http://www.sciencedirect.com/science/article/pii/B9780128129708000105, doi:https://doi.org/10.1016/B978-0-12-812970-8.00010-5.
Kumar, P., Khani, A., He, Q., 2018. A robust method for estimating transit passenger trajectories using automated data. Transportation Research Part C: Emerging Technologies 95, 731–747.
Kusakabe, T., Iryo, T., Asakura, Y., 2010. Estimation method for railway passengers' train choice behavior with smart card transaction data. Transportation 37, 731–749.
Li, W., 2014. Route and transfer station choice modeling in the mtr system. Working paper.
Liu, Z., Wang, S., Chen, W., Zheng, Y., 2016. Willingness to board: a novel concept for modeling queuing up passengers. Transportation Research Part B: Methodological 90, 70–82.
Mo, B., Ma, Z., Koutsopoulos, H., Zhao, J., 2020a. Calibrating route choice for urban rail system: A comparative analysis using simulation-based optimization methods, Transportation Research Board 99th Annual Meeting.
Mo, B., Ma, Z., Koutsopoulos, H., Zhao, J., 2020b. Capacity-constrained network performance model for urban rail systems. Transportation Research Record.
Nelder, J.A., Mead, R., 1965. A simplex method for function minimization. The computer journal 7, 308–313.
Osorio, C., Bierlaire, M., 2013. A simulation-based optimization framework for urban transportation problems. Operations Research 61, 1333–1345.
Prato, C.G., 2009. Route choice modeling: past, present and future research directions. Journal of choice modelling 2, 65–100.
Preston, J., Pritchard, J., Waterson, B., 2017. Train overcrowding: investigation of the provision of better information to mitigate the issues. Transportation research record 2649, 1–8.
Regis, R.G., Shoemaker, C.A., 2005. Constrained global optimization of expensive black box functions using radial basis functions. Journal of Global optimization 31, 153–171.
Scipy, 2019. Scipy dual annealing algorithm. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.dual_annealing.html.
Snoek, J., Larochelle, H., Adams, R.P., 2012. Practical bayesian optimization of machine learning algorithms, in: Advances in neural information processing systems, pp. 2951–2959.
Spall, J.C., et al., 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE transactions on automatic control 37, 332–341.
Sun, L., Lu, Y., Jin, J.G., Lee, D.H., Axhausen, K.W., 2015. An integrated bayesian approach for passenger flow assignment in metro networks. Transportation Research Part C: Emerging Technologies 52, 116–131.
Sun, Y., Xu, R., 2012. Rail transit travel time reliability and estimation of passenger route choice behavior: Analysis using automatic fare collection data. Transportation Research Record 2275, 58–67.
Appendices

Appendix A. Passenger path Choice Model for MTR System

These results are from (Li, 2014). The C-logit Model formulation is the same as Eq. (1) and Eq. (2). A total number of 31,640 passengers completed the questionnaire. After filtering duplicate responses, 26,996 responses were available. The model results are shown in Table A.8. The main explanatory variables are the total in-vehicle time, relative transfer walking time, and the number of transfers. All variables are statistically significant with the expected signs. Paths with high in-vehicle time, walking time, and the number of transfers are less likely to be chosen by passengers.

|                          | Estimate | Std. Error | t-value |
|--------------------------|----------|------------|---------|
| In-vehicle time          | -0.147   | 0.011      | -13.64  *** |
| Relative walking time    | -1.271   | 0.278      | -4.56   *** |
| Number of transfers      | -0.573   | 0.084      | -6.18   *** |
| Commonality factor       | -3.679   | 1.273      | -2.89   ** |

\[ p^2 = 0.54 \]

***: \( p < 0.01 \); **: \( p < 0.05 \).