

## A Comparison of Different Approaches to Dynamic Origin-Destination Matrix Estimation in Urban Traffic

Andersen, Nicklas Sindlev; Chiarandini, Marco; Debrabant, Kristian

*Publication date:*  
2022

*Document version:*  
Submitted manuscript

*Document license:*  
CC BY

*Citation for published version (APA):*  
Andersen, N. S., Chiarandini, M., & Debrabant, K. (2022, May 31). A Comparison of Different Approaches to Dynamic Origin-Destination Matrix Estimation in Urban Traffic. arXiv.

Go to publication entry in University of Southern Denmark's Research Portal

### **Terms of use**

This work is brought to you by the University of Southern Denmark.  
Unless otherwise specified it has been shared according to the terms for self-archiving.  
If no other license is stated, these terms apply:

- You may download this work for personal use only.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying this open access version

If you believe that this document breaches copyright please contact us providing details and we will investigate your claim.  
Please direct all enquiries to [puresupport@bib.sdu.dk](mailto:puresupport@bib.sdu.dk)

# A Comparison of Different Approaches to Dynamic Origin-Destination Matrix Estimation in Urban Traffic

Nicklas Sindlev Andersen<sup>a</sup>, Marco Chiarandini<sup>a</sup> and Kristian Debrabant<sup>a</sup>

<sup>a</sup>University of Southern Denmark (SDU), 55 Campusvej, Odense M, Denmark.

## ARTICLE HISTORY

Compiled June 2, 2022

## ABSTRACT

Given the counters of vehicles that traverse the roads of a traffic network, we reconstruct the travel demand that generated them expressed in terms of the number of origin-destination trips made by users. We model the problem as a bi-level optimization problem. At the inner-level, given a tentative demand, we solve a Dynamic Traffic Assignment (DTA) problem to decide the routing of the users between their origins and destinations. Finally, we adjust the number of trips and their origins and destinations at the outer-level to minimize the discrepancy between the counters generated at the inner-level and the given vehicle counts measured by sensors in the traffic network. We solve the DTA problem by employing a mesoscopic model implemented by the traffic simulator SUMO. Thus, the outer problem becomes an optimization problem that minimizes a black-box Objective Function (OF) determined by the results of the simulation, which is a costly computation. We study different approaches to the outer-level problem categorized as gradient-based and derivative-free approaches. Among the gradient-based approaches, we look at an assignment matrix-based approach and an assignment matrix-free approach that uses the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm. Among the derivative-free approaches, we investigate Machine Learning (ML) algorithms to learn a model of the simulator that can then be used as a surrogate OF in the optimization problem. We compare these approaches computationally on an artificial network. The gradient-based approaches perform the best in terms of solution quality and computational requirements. In contrast, the results obtained by the ML approach are currently less satisfactory but provide an interesting avenue for future research.

## KEYWORDS

Bi-level optimization; Dynamic origin-destination matrix estimation; Dynamic traffic assignment; Simulation-based optimization

## 1. Introduction

Traffic on the road networks of urban areas is continuously increasing. Municipalities can act on the infrastructure to keep high mobility standards, modifying it to avoid congestion and ensure reliability. Therefore, there is an urge to provide decision-makers with tools to make informed decisions. For this purpose, traffic simulation software has been developed to assess different planning scenarios. To use these simulators, the *network supply*, and the *travel demand* must be known with some certainty. The network supply is, in general, defined as the maximum number of vehicles the network

road infrastructure can handle at a given time. In contrast, the demand can generally be said to be the number of vehicles that would like to travel on the network at a given time.

The network supply and travel demand interact dynamically, i.e., the users travel in their vehicles from one location to another on the network and interact with each other and with the road infrastructure. However, congestion may occur, and users might change their decisions and re-distribute on the network. If the demand and the network supply are known, it can be possible to obtain accurate distributions of users on the given network by calculating an equilibrium situation corresponding to solving a *Dynamic Traffic Assignment (DTA)* problem. Then, by varying the network supply, one can analyze different scenarios to address a network design task. Consequently, if the demand in a particular traffic network is unknown, it is desirable to estimate it.

Travel demand can be modeled in many different ways but is commonly modeled using *dynamic Origin-Destination (OD) matrices*. These matrices describe the aggregate demand pattern changing over time (dynamic) in a set of time intervals of a much larger analysis period. Due to their simplicity and conciseness, these dynamic OD matrices are usually the required input of many traffic simulation software packages. Because of this, the problem of trying to estimate dynamic OD matrices has gained a lot of attention from researchers in recent years.

The problem of estimating dynamic OD matrices is usually referred to as the *Dynamic OD Estimation (DODE)* problem in the literature. Different formulations exist, but it is commonly formulated as a bi-level optimization problem consisting of an inner and an outer optimization problem. A proposed dynamic OD matrix is given as input to the inner problem, and a DTA problem is solved. The result is a routing of the users in the network between origins and destinations at every time interval. In the outer optimization problem, the dynamic OD matrices are adjusted to minimize the discrepancy between the routing returned by the solution of the inner problem and the one observed in real-life, i.e., the number of vehicles that transited in every time interval on every road of the network as measured by sensors. The observed routing is usually expressed in terms of the number of vehicles, average speeds, densities, etc., that have been captured by sensors located along roads in the traffic network under investigation. However, other kinds of traffic observations can also be included (e.g., travel times, intersection turning ratios, etc.). This information is assumed to be directly related to the unknown travel demand and can guide the retrieval of such a demand. In other words, the DODE problem can be summarized as the problem of finding the dynamic OD matrices that, when assigned to the traffic network as vehicles traveling between origins and destinations, reproduce the traffic measurements captured by sensors in the network. In addition to this, the dynamic OD matrices should be close to some hypothesized ones that can be determined, e.g., by the number of people living in the zones that are considered as origins and destinations.

Solving bi-level optimization problems is quite challenging, as the inner problem constrains the outer problem, i.e., only an optimal solution to the inner problem is a feasible solution to the outer problem. To overcome this challenge, it is crucial to design solution approaches that are efficient and able to produce good and reliable solutions in a reasonable amount of time. As such, much of the recent research on the DODE problem tries to include and leverage additional input data or problem-specific knowledge to improve the efficiency of the applied algorithms.

The DODE problem is solved frequently for a certain geographical area of study and traffic network whenever a more up-to-date demand estimate is needed. If the general structure of the traffic network and the possible OD pairs within the area of

study remain the same over a reasonable amount of time, the input data in the form of recent traffic observations is the only component of the DODE problem that changes from one optimization task to another. The solution approaches that apply sequential optimization algorithms to solve the DODE problem imply performing many DTAs in each separate optimization task, essentially starting from scratch every time a new demand estimate is needed. In a single optimization task, vast computational resources are spent on performing the DTAs, and each DTA produces valuable information. This information is usually discarded and not used in subsequent optimization tasks, primarily due to the inherent sequential nature of some of the applied solution approaches. In this context, an ML approach becomes appealing as it allows leveraging data produced by many DTAs.

Therefore, we study an approach to bypass the computationally costly and time-consuming computations performed in the solution of the inner optimization problem by learning a model for the input-output relationship established by solving this inner problem.

To benchmark the ML approaches, we compare them combined with classical gradient-based approaches. In addition, we provide an analysis of the advantages and disadvantages of these approaches.

Our work provides the following contributions: We design a possible way to use ML in the solution of the DODE problem by learning a model for the inner problem and reducing its computational cost. Further, we perform a fair computational comparison under a controlled scenario, carefully designed and synthetically generated, of two state-of-the-art methods and our new ML approach.

Finally, we provide an analysis of the advantages and disadvantages of the different approaches and indications about the contexts in which they could be performing the best.

All methods are here reimplemented and share the same subroutines. We use the well-established, open-source traffic simulator SUMO (Lopez et al. 2018) to perform the DTA in the inner optimization problem. As such, the DTA performed by SUMO will be regarded as a black-box function that assigns to a given input dynamic OD matrix a set of path flows in the network according to a certain equilibrium criterion called a Stochastic User Equilibrium (SUE). This equilibrium criterion is described in more detail in Section 3.

The rest of the chapter is outlined as follows. First, Section 2 gives a general overview of the existing literature on the DODE problem. Then, Section 3 introduces notation and the traditional (continuous) bi-level optimization formulation of the DODE problem that we will use in the subsequent sections. In Section 4, we describe the approaches considered in the study, and in Section 5, the irregular grid network and the synthetic problem data that we used for our experiments. Next, the results are presented in Section 6. The best method turns out to be a gradient-based approach with assignment matrix modeling, while the ML approach shows inferior performance. Finally, we conclude with a discussion of the results in Section 7.

## 2. Literature Review

In the classical static OD estimation problem, static OD trips are considered. The problem is to estimate the demand in a single time interval (a single OD matrix is estimated). On the other hand, time-dependent OD trips are considered in the DODE problem, and the demand can vary in the different time intervals (i.e., time-dependent

OD matrices are estimated).

The static OD matrix estimation problem has a long history, and a large body of literature on this problem is available. The ideas and insights that have been gained through studying the static problem are also extended and applied to the dynamic problem. For an overview of the different classical solution approaches to the static problem, see [Abrahamsson \(1998\)](#); [Marzano, Papola, and Simonelli \(2009\)](#); [Bert \(2009\)](#). On the other hand, the most recent and extensive literature review on the dynamic problem is provided in [Omriani and Kattan \(2012\)](#); [Djukic \(2014\)](#). In contrast, a literature review on much earlier work is provided in [Balakrishna \(2006\)](#). For a more general overview of the DODE problem and the different modeling components and decisions that have to be considered when solving this problem, we refer the reader to [Lindveld \(2003\)](#); [Bert \(2009\)](#). Finally, an overview of related and relevant problems is provided in [Antoniou et al. \(2016\)](#).

Most of the solution approaches to the DODE problem described in the literature can be divided into two broad classes: DTA-based and non-DTA-based approaches. A DTA-based approach is usually used in congested networks where the path choices of the users of the network and the road-use patterns are of interest. DTA-based approaches provide a sound way to consider behavioral factors that affect travel decisions but are more computationally demanding than non-DTA-based approaches. Contrary to this, in a non-DTA-based approach, vehicles are assumed to take the shortest path, which is usually also the one that minimizes the travel time. In this case, if the users of the traffic network are assumed to be able to take the shortest path, then to solve the DODE problem, a non-DTA-based approach is usually sufficient. Therefore, we focus on the literature concerned with DTA-based solution approaches with this distinction in mind.

Usually, solution approaches to the outer problem of the bi-level formulation of the DODE problem consider the problem of minimizing the discrepancy between the quantities determined by a DTA solution and the corresponding observed quantities given as input. The problem is treated as a continuous optimization problem, and general-purpose optimization algorithms from local and global optimization are used. The main advantage of general-purpose algorithms is that they do not require exact knowledge of the functional relationship of the variables to be estimated. The drawback is that these algorithms are usually inefficient in their original form if the different algorithm components and parameters are not adapted and tuned to the problem at hand. The general-purpose algorithms applied to the DODE problem can be categorized as either *gradient-based* or *derivative-free* optimization algorithms. Gradient-based algorithms are iterative procedures that adjust estimates based on the information provided by the gradient and possibly higher-order derivatives. This is not the case for derivative-free algorithms (e.g., simulated annealing, evolutionary algorithms, and sampling-based algorithms, such as the Nelder-Mead algorithm) that usually rely solely on Objective Function (OF) evaluations.

We can further identify *assignment matrix-based* or *assignment matrix-free* approaches within the class of gradient-based algorithms. The distinction can be made based on whether an Assignment Matrix (AM) is used as a part of the applied optimization algorithm or not. An AM summarizes the result of a DTA and consists of elements that describe the utilization of the roads in the traffic network with respect to the number of users that travel between origins and destinations within a certain time period. This means that the AM varies as a function of the travel demands. The solution approaches to the DODE problem that uses gradients and AMs exploit that it can be possible to analytically derive the exact gradient from a functional relationship

between the variables through the AM. On the other hand, a gradient-based but AM-free approach primarily uses finite difference approximations of the gradient, which are established through OF evaluations.

The most recent studies that adopt a gradient and AM-based approach are given in Bert (2009); Toledo and Kolechkina (2013); Frederix, Viti, and Tampère (2013); Djukic et al. (2017); Shafiei et al. (2017); Masip et al. (2018). These studies primarily focus on improving the efficiency of the applied algorithms by better modeling the functional relationship between the variables that are to be estimated. A common theme for the gradient and AM-based approaches is that count observations are the only type of traffic observations usually given as input to the optimization problem. The modeling of the relationship between the variables to be estimated can be done through the AM. However, using this type of approach limits the type of observations that can be accommodated in the optimization problem. For example, the functional relationship between demands and count observations can easily be established if a proportional relationship between these variables is assumed (see Eq.9). However, for other types of observations, it can be harder and more cumbersome to determine appropriate relationships between the variables to be estimated.

Gradient and AM-free approaches have also gained considerable attention in the literature. Several researchers have especially been interested in studying and applying different variations of the Stochastic Perturbation Simultaneous Approximation (SPSA) algorithm by Spall (Spall 1998), tailoring it to the DODE problem. This algorithm only relies on OF evaluations with no need for an explicit characterization of the relationship between the variables to be estimated. In each iteration of the SPSA algorithm, OF evaluations are used to approximate the gradient. This approach makes it possible to include other types of traffic observations besides count observations, and it has therefore been the predominant approach in these cases (Vaze et al. 2009; Cipriani et al. 2011; Tympakianaki, Koutsopoulos, and Jenelius 2015; Antoniou et al. 2015; Lu et al. 2015; Carrese et al. 2017). Just like most gradient-based optimization algorithms, the performance of the SPSA algorithm is sensitive to (i) the tuning of algorithm parameters, (ii) the possibly very different magnitudes of the variables to be estimated, and (iii) the OF shape. Different enhancements to the SPSA algorithm in the mentioned references thus have also been focused on components that address these points to improve the stability and robustness of the algorithm when applied to the DODE problem.

A *surrogate model* can be estimated based on empirical data if a model for the OF is not available through theoretical argumentation. Gradient-based or derivative-free methods can then be applied using the surrogate model. In Balakrishna (2006), a model-based and derivative-free method is compared against the SPSA algorithm. Response surface techniques are used to model the OF by low order polynomials fitted locally to the values in correspondence of sample points of the search space. The author applies a Stable Noisy Optimization by Branch and Fit (SNOBFIT) algorithm that uses a derivative-free Box-Complex algorithm. The Box-Complex algorithm is a direct search and sampling-based method that extends the well-known Nelder-Mead algorithm. The Box-Complex and SNOBFIT algorithm performed worse in efficacy and scalability than the SPSA algorithm.

Generally, the derivative-free solution algorithms require a high number of OF evaluations to obtain results comparable to the gradient-based solution algorithms (Audet and Hare 2017).

A promising line of research is provided in Osorio (2019), which describes a surrogate model-based approach that relies on fitting simplified analytical models to the

underlying simulation and DTA model resulting in a system of equations that can be solved efficiently by standard system of equations solvers. The approach thus obtains good computational results under tight computational budgets. The surrogate model-based approach is compared against the SPSA and a derivative-free pattern search algorithm. The approach performs well, while the pattern search algorithm and the SPSA algorithm perform poorly in comparison.

Our work contributes to this thread of research by studying ML approaches to determine surrogate models as an alternative to response surface techniques [Balakrishna \(2006\)](#) and the surrogate model-based approach proposed in [Osorio \(2019\)](#).

Fewer studies can be found where model-free (and hence derivative-free) algorithms have been applied to the DODE problem. These algorithms only rely on OF evaluations and exhibit the advantage that they can be distributed and parallelized, as they allow independent OF evaluations. Evolutionary algorithms have been the most popular among derivative-free optimization algorithms. Notably, in [Kattan and Abdulhai \(2006\)](#), an evolutionary algorithm was applied in a setting with distributed and parallel computing. Other studies that apply evolutionary algorithms can be found in [Vaze et al. \(2009\)](#); [Omrani \(2014\)](#), while [Tsekeris, Dimitriou, and Stathopoulos \(2007\)](#) also use an evolutionary algorithm, but in a non-DTA based approach. However, in [Vaze et al. \(2009\)](#), the evolutionary algorithm was shown to perform worse than the SPSA algorithm.

Two variants of DODE problems are usually addressed in the literature. Their offline or online setting characterizes them. In an offline setting, historical data is used to estimate the network-wide demand, such that long-term predictions of future traffic conditions can be made. This is especially beneficial to decision-makers. It enables them to evaluate different planning scenarios and make informed decisions when proposed changes to infrastructure and facilities are made. Offline demand estimation procedures are usually applied in network-wide studies, so the computational cost can be considerably high in this setting. In an online setting, real-time traffic data is used and historical data to estimate the current level of demand such that short-term predictions of future traffic conditions can be made. This primarily benefits real-time traffic control and path-guidance systems. Online demand estimation procedures are usually applied on smaller traffic networks or locally, e.g., at intersections, where traffic patterns are identified, such that adaptive traffic control strategies can be used. For references on work that studies the online DODE problem, we refer the reader to [Ashok and Ben-Akiva \(2000\)](#); [Antoniou et al. \(2009\)](#); [Omrani and Kattan \(2012\)](#). In this work, we focus on the offline, DTA-based, bi-level formulation of the DODE problem.

### 3. Notation and Problem Statement

A traffic network can be defined as a directed graph  $G = (N, A)$  consisting of a set of nodes  $N$  and a set of arcs  $A$ . Each node in the network represents a junction or an origin/destination point, while arcs represent roads. The arcs are directed, which means that the traffic between two nodes can be uni-directional and one-way roads are possible. The arcs in the network have additional attributes that generally indicate how much and how well the roads in the network can handle traffic. Some examples of essential arc attributes are the number of lanes, the length (kilometer) and a free-flow speed (kilometer/hour) that is the maximum allowed speed a vehicle can travel with on an uncongested road. Additionally, geographical information can be added to the arcs

and nodes to indicate their physical location and shape.

A subset of the nodes  $N$  in a traffic network can be identified as possible origins and/or destinations. More precisely, we let  $O \subseteq N$  be a set of origins and  $D \subseteq N$  a set of destinations, where it is usually the case that  $D \cap O \neq \emptyset$ , i.e., it is possible for a node to be an origin and a destination simultaneously. A *trip* is the movement of a vehicle from one location to another. More precisely, a trip departs from an origin  $i \in O$  and terminates at a destination  $j \in D$ . In this case, we associate the trip with an OD  $w = (i, j) \in W = O \times D$ . Here  $W$  is the set of all OD pairs with size  $|W| = m$ , and it is assumed that no trip departs and arrives in the same location, i.e., the OD pairs are defined such that  $i \neq j$  for all  $w = (i, j) \in W$ . Finally, a subset of the arcs in the network  $G$  is assumed to be equipped with sensors defined by the set  $Q = \{1, \dots, n_Q\} \subseteq A$ .

To define the quantities of interest in the DODE problem, we introduce an analysis period  $T = [0, t_{\text{end}})$  discretized into a set of  $n_S$  subintervals  $\{[0, t_1), [t_1, t_2), \dots, [t_{n_S-1}, t_{n_S} = t_{\text{end}})\}$  of equal duration. For the sake of convenience, we identify these intervals by the indices in the set  $S = \{1, \dots, n_S\}$ . These indices define the time intervals in which we want to estimate the demands for each of the  $m$  OD pairs. The estimation of the demands is then based on the arc count observations made within these time intervals.

The quantities of interest are usually referred to as being arranged in matrices. Here, however, we will arrange these quantities in vectors. In other words, we flatten or vectorize the matrices, i.e., given a matrix  $\mathbf{A} \in \mathbb{R}_+^{p_1 \times p_2}$ , we concatenate consecutive rows of the matrix in a column vector ( $\mathbb{R}^{p_1 \times p_2} \rightarrow \mathbb{R}^{p_1 \cdot p_2}$ ):

$$\text{vec}(\mathbf{A}) = \mathbf{a} = [a_{11}, \dots, a_{p_1 1}, \dots, a_{12}, \dots, a_{p_1 2}, \dots, a_{p_1 p_2}]^\top \in \mathbb{R}^{p_1 \cdot p_2}. \quad (1)$$

The quantities of interest can then be described in terms of the following vectors:

- $\mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}_+^{m \cdot n_S}$  are the vectors of *estimated demands* and *seed demands*, respectively. Each element of these vectors  $x_{ws}$  or  $\tilde{x}_{ws}$  is associated with an OD pair  $w \in W$  and a time interval  $s \in S$ . The vector  $\mathbf{x}$  of estimated demands contains the estimate of the number of trips made for each OD pair in each of the time intervals. The vector  $\tilde{\mathbf{x}}$  of seed demands defines the prior knowledge of the number of trips made for each OD pair in each time interval. This prior knowledge is usually assumed to have been obtained from a previous demand study, e.g., from a population survey. Note that we model discrete values as real numbers. We thus define the continuous relaxation of a discrete optimization problem.
- $\mathbf{x}^{\text{Lower}}, \mathbf{x}^{\text{Upper}} \in \mathbb{R}_+^{m \cdot n_S}$  are the vectors of lower and upper bounds on the estimated demands, respectively. These upper and lower bounds define the *search space*.
- $\mathbf{c}, \hat{\mathbf{c}} \in \mathbb{Z}_+^{n_Q \cdot n_S}$  are the vectors of estimated and observed arc counts, respectively. Each element of these vectors  $c_{qs}$  or  $\hat{c}_{qs}$  is associated with a sensor  $q \in Q$  and a time interval  $s \in S$ .

The outer optimization problem of the bi-level formulation of the DODE problem can now be defined as:

$$\min F(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{c}(\mathbf{x}), \hat{\mathbf{c}}) \quad (2)$$

$$\text{subject to } \mathbf{x}^{\text{Lower}} \leq \mathbf{x} \leq \mathbf{x}^{\text{Upper}} \quad (3)$$



The solution of this problem,  $\mathbf{x}^*$ , is the demand estimate that results from minimizing the OF  $F$ , which we define as the weighted sum of the measures of discrepancy between the estimated quantities and their corresponding observed or a priori values:

$$F(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{c}(\mathbf{x}), \hat{\mathbf{c}}) = \omega_1 \cdot f^{(1)}(\mathbf{x}, \tilde{\mathbf{x}}) + \omega_2 \cdot f^{(2)}(\mathbf{c}(\mathbf{x}), \hat{\mathbf{c}}), \quad \omega_1, \omega_2 \in \mathbb{R}_+. \quad (4)$$

In our specific case, the discrepancy  $f^{(1)}$  is measured between the estimated demands  $\mathbf{x}$  and a priori known demands  $\tilde{\mathbf{x}}$  and the discrepancy  $f^{(2)}$  between the estimated arc counts  $\mathbf{c}(\mathbf{x})$  and the observed arc counts  $\hat{\mathbf{c}}$ . Note that in a more general setting, the OF may consist of several additional terms  $f^{(3)}, f^{(4)}, \dots$  with respective weights  $\omega_3, \omega_4, \dots$  that consider additional available information, such as arc speed, density, travel times, turning ratios, etc., that can be included to improve the guidance of the search process.

An inner optimization problem must be solved to evaluate the OF in Eq. (2) for a proposed vector  $\mathbf{x}$  of demands to find a corresponding vector  $\mathbf{c}(\mathbf{x})$  of arc counts. The inner optimization problem takes the form of a DTA problem. The task is to determine the routing of users between origins and destinations according to a particular assignment principle. The users of the traffic network are assumed to make decisions per the criterion specified by the assignment principle. Two examples of assignment principles that are widely used and studied within the area of transportation research are (i) the User Equilibrium (UE) principle (Wardrop’s first principle (Friesz and Bernstein 2016)), which states that each user makes decisions that minimize their own individual travel time, and (ii) the System Optimum (SO) principle (Wardrop’s second principle (Friesz and Bernstein 2016)), which states that users make joint decisions to minimize the total system travel time.

DTA models adopt an assignment principle and incorporate several different travel choice components to reflect the travel choices a user of an actual traffic network might face when wanting to travel between an origin and a destination. Travel choices that are possible to model and include in a DTA model are, among others: path choice, departure time choice, mode choice, and destination choice, to name a few. However, in the context of the DODE problem, a simple DTA model that only incorporates path and departure time choice components is usually used, meaning path and departure time choices are endogenous to the DTA model, while other travel choice components are exogenous or fixed.

To obtain accurate travel times between origins and destinations, a DTA model relies on an underlying traffic flow model, i.e., detailed traffic flow models are used by DTA models to explicitly propagate vehicles from one arc to another while considering space constraints. In other words, queuing and congestion are modeled, and the effects of these phenomena are reflected in the travel time. For a good introduction and a general overview of the DTA problem, we refer the reader to Peeta and Ziliaskopoulos (2001), as we only give a brief description here.

Two traffic flow models are implemented in SUMO and can be used in conjunction with a DTA: a *mesoscopic* and a *microscopic*. These two types of traffic flow models differ in the level of detail in which they model traffic flow dynamics. In the mesoscopic model, the arcs in a traffic network are modeled as discrete queues through which vehicles are propagated. Further, a coarser model for intersections and lane-changing is used (German Aerospace Center 2022). On the other hand, in the microscopic model, detailed quantities such as position, speed, acceleration, and braking distance are considered in the propagation of vehicles on an arc.

The mesoscopic model is computationally cheaper to solve and can provide the necessary data to a sufficient degree of detail needed in the estimation problem. Therefore,

we choose that model.

A DTA can be obtained heuristically through an iterative simulation process using the mesoscopic traffic flow model implemented in SUMO. At the end of the iterative process, a stationary distribution of the path choice decisions of the users of the traffic network respects an assignment principle. This assignment principle is the Stochastic User Equilibrium (SUE) in SUMO. The SUE assignment principle extends the UE assignment principle, where stochastic elements have been incorporated into the DTA model. This form of UE assignment is regarded as the more realistic, as uncertainty is incorporated in the assignment model to take into account the uncertainty of the users' knowledge about network conditions. Contrary to the UE assignment principle, which is defined in terms of *actual* travel times, the SUE assignment principle is instead defined in terms of *perceived* travel times.

The iterative method used by SUMO to determine a DTA that respects the SUE assignment principle is described in [Gawron \(1998\)](#). This method determines the probabilities of choosing between path alternatives for each user who wants to travel between an origin and destination at a certain time. In brief, the path choice probabilities are determined based on (i) the arc travel times experienced in the previous iteration, (ii) the sum of arc travel times along different least-cost paths (these paths constitute a set of alternatives to a user), and (iii) the previous probabilities of choosing the paths. These quantities are used to obtain new estimates of the path choice probabilities at each iteration. Finally, we note that the least-cost paths are computed in parallel in SUMO by a time-dependent version of Dijkstra's algorithm, while a single traffic simulation is single-threaded and sequential.

Note that we modeled demands  $\mathbf{x}$  as continuous variables while the mesoscopic simulator implemented by SUMO solves a discrete optimization problem. When the demands are given to SUMO as input, they are rounded to the nearest integer values. In all other cases, the demands are handled as being continuous, which allows us to apply continuous optimization algorithms in the outer-level problem.

For our purposes here, the DTA performed by SUMO can conveniently be defined by the function:

$$\Gamma : \mathbb{R}_+^{m \cdot n_s} \rightarrow \mathbb{Z}_+^{n_Q \cdot n_s}. \quad (5)$$

This function maps demands  $\mathbf{x}$  to arc counts through a DTA, i.e.,  $\mathbf{x} \mapsto \Gamma(\mathbf{x}) = \mathbf{c}(\mathbf{x})$ . The arc counts  $\mathbf{c}(\mathbf{x})$ , together with the corresponding input demands  $\mathbf{x}$ , can then be used in the OF in Eq. (4) to determine the quality of the current demand estimate.

Eq. (3) defines upper and lower bounds on the demands. These are usually supplied to avoid a situation where an unrealistic high amount of demand is loaded on the traffic network, exceeding the network's capacity. In a realistic setting, the bounds on the demands are usually determined based on population survey data and experimental results from previous studies.

In the remaining part of this work, we use the shorthand notation  $F(\mathbf{x})$  for the OF defined in Eq. (4), as  $\tilde{\mathbf{x}}$  and  $\hat{\mathbf{c}}$  are given and  $\mathbf{c}(\mathbf{x})$  is derived from  $\mathbf{x}$ .

#### 4. Solution Approaches

Different definitions of the discrepancy between observed and estimated quantities in the OF  $F$  are possible. Drawing on the knowledge obtained in [Cipriani et al. \(2011\)](#) and [Antonioni et al. \(2016\)](#), we use an OF that penalizes large errors between observed

and estimated quantities and that weights the different terms evenly, i.e., we take into account that the different quantities  $f^{(1)}$  and  $f^{(2)}$  that enter into the OF can have different magnitudes and hence normalize their values. Thus, we define the terms  $f^{(1)}$  and  $f^{(2)}$  used in the OF  $F$  in the following functional form:

$$f^{(1)}(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{\sqrt{\sum_{\substack{w \in W \\ s \in S}} (x_{ws} - \tilde{x}_{ws})^2}}{\sqrt{\sum_{\substack{w \in W \\ s \in S}} \tilde{x}_{ws}^2}} \quad \text{and} \quad f^{(2)}(\mathbf{c}(\mathbf{x}), \hat{\mathbf{c}}) = \frac{\sqrt{\sum_{\substack{q \in Q \\ s \in S}} (c_{qs} - \hat{c}_{qs})^2}}{\sqrt{\sum_{\substack{q \in Q \\ s \in S}} \hat{c}_{qs}^2}}. \quad (6)$$

To further limit the search space, we introduce generation constraints (Antoniou et al. 2016) that set an upper bound on the number of outbound trips from an origin. we let  $o_i$  denote the maximum number of vehicles that can leave origin  $i \in O$  and enforce that:

$$\sum_{\substack{j \in D, \\ w=(i,j) \in W \\ s \in S}} x_{ws} \leq o_i \quad \forall i \in O. \quad (7)$$

This type of constraint can easily be incorporated into the problem formulation as it is typically the case that data for the values of  $o_i$ ,  $i \in O$  are readily available and highly reliable, e.g., from survey data or population density statistics. Other possible constraints are also mentioned in Antoniou et al. (2016), such as trip distribution constraints or attraction constraints that, contrary to the generation constraints, limit the number of trips that can arrive at a certain destination. However, data pertaining to these constraints might be harder to provide and will thus not be considered in this study.

The algorithms that follow are iterative algorithms. We define an iteration index  $\tau \in \{0, \dots, \tau_{\max}\}$  where  $\tau_{\max}$  is the maximum number of iterations of an algorithm. We thus let  $\mathbf{x}_\tau$  define the estimate at iteration  $\tau$ .

#### 4.1. Gradient-based Approaches

Gradient-based algorithms for continuous optimization problems take the general form of an iterative procedure where an incumbent estimate at iteration  $\tau$  is adjusted in such a way that it is moved towards a minimum to yield the new estimate at iteration  $\tau + 1$ :

$$\mathbf{x}_{\tau+1} = \mathbf{x}_\tau + \eta_\tau \cdot \mathbf{d}_\tau. \quad (8)$$

Here  $\eta_\tau$  is a scalar value that we will refer to as the *step rate*, and  $\mathbf{d}_\tau$  is the *descent direction*. The descent direction is determined by the gradient through  $\mathbf{d}_\tau = -\mathbf{g}_\tau$ . Different gradient-based optimization algorithms differ in how the descent direction and the step rate are chosen. For the DODE problem, it is possible to distinguish two ways to derive the gradient: AM-based and AM-free.

#### 4.1.1. An Assignment Matrix-based Approach

Through the use of an AM the gradient of the OF can be calculated analytically. The (vectorized) AM  $\mathbf{p}_\tau = \text{vec}(\mathbf{P}_\tau(\mathbf{x}_\tau))$  at an iteration  $\tau$  consists of entries  $p_{qrs} \in [0, 1]$  that describe the proportion of trips  $x_{ws}$  for OD pair  $w \in W$  in time interval  $s \in S$  that go through arc  $q \in Q$  in time interval  $r \in S$ ,  $r \geq s$ . In other words, depending on a number of factors such as the travel time between origins and destinations and the length of the time intervals  $s \in S$ , the traffic present on an arc  $a \in A$  may originate from OD trips from several different OD pairs from the current time interval or several previous time intervals. Using this knowledge the OD trips can be related to the arc counts through the linear relation:

$$c_{qr} = \sum_{\substack{w \in W \\ s \in S \\ s \leq r}} p_{qrs} \cdot x_{ws}, \quad \forall q \in Q, r \in S. \quad (9)$$

Through the use of the AM it is possible to derive an analytical expression for the gradient of the OF  $F$  at every iteration:

$$\frac{\partial F}{\partial \mathbf{x}} = \omega_1 \cdot \mathbf{g}^{(1)} + \omega_2 \cdot \mathbf{g}^{(2)}, \quad (10)$$

where the vectors  $\mathbf{g}^{(1)}$  and  $\mathbf{g}^{(2)}$  are the gradients of the two terms  $f^{(1)}$  and  $f^{(2)}$ , respectively, whose elements can be derived to be (see A):

$$g_{ws}^{(1)} = \frac{\partial f^{(1)}}{\partial x_{ws}} = \frac{x_{ws} - \tilde{x}_{ws}}{\sqrt{\sum_{\substack{w' \in W \\ s' \in S}} (x_{w's'} - \tilde{x}_{w's'})^2} \cdot \sqrt{\sum_{\substack{w' \in W \\ s' \in S}} \tilde{x}_{w's'}^2}}, \quad \forall w \in W, s \in S, \quad (11)$$

$$g_{ws}^{(2)} = \frac{\partial f^{(2)}}{\partial x_{ws}} = \frac{\sum_{\substack{q' \in Q \\ r' \in S \\ r' \geq s}} (c_{q'r'} - \hat{c}_{q'r'}) \cdot p_{q'r'ws}}{\sqrt{\sum_{\substack{q' \in Q \\ r' \in S}} (c_{q'r'} - \hat{c}_{q'r'})^2} \cdot \sqrt{\sum_{\substack{q' \in Q \\ r' \in S}} \hat{c}_{q'r'}}}, \quad \forall w \in W, s \in S. \quad (12)$$

The descent direction in Eq. (8) is then simply the negative of the expression in Eq. (10):

$$\mathbf{d}_\tau = - \left( \omega_1 \cdot \mathbf{g}_\tau^{(1)} + \omega_2 \cdot \mathbf{g}_\tau^{(2)} \right). \quad (13)$$

#### 4.1.2. An Assignment Matrix-free Approach

As mentioned in Section 2, most of the studied gradient-based and AM-free approaches for the DODE problem use the SPSA algorithm. This algorithm uses the negative of the gradient as a way to obtain a descent direction in a similar way as the previous approach. As no AM is needed in the derivation of the gradient of the first OF term  $f^{(1)}$ , we can simply use the exact gradient defined in Eq. (11). However, the gradient

of the second OF term  $f^{(2)}$  will have to be approximated because now no AM is given and no functional relationship between the variables to be estimated is assumed.

A one-sided finite difference approximation of the gradient is usually used:

$$\tilde{\mathbf{g}}_\tau^{(2)} = \frac{f^{(2)}(\mathbf{c}(\mathbf{x}_\tau), \hat{\mathbf{c}}) - f^{(2)}(\mathbf{c}^-(\mathbf{x}_\tau), \hat{\mathbf{c}})}{b_\tau} \cdot \overline{\mathbf{\Delta}}_\tau, \quad \tau \in \{0, \dots, \tau_{\max}\}. \quad (14)$$

Here,  $\mathbf{c}(\mathbf{x}_\tau) = \Gamma(\mathbf{x}_\tau)$ ,  $\mathbf{c}^-(\mathbf{x}_\tau) = \Gamma(\mathbf{x}_\tau - b_\tau \cdot \mathbf{\Delta}_\tau)$  and  $\mathbf{\Delta}_\tau \in \{-1, 1\}^{m \cdot n_s}$  is a random perturbation vector, where the entries  $\Delta_{ws}$ ,  $w \in W$ ,  $s \in S$  are independent Bernoulli random variables taking the value  $\pm 1$  with probability  $1/2$ . Furthermore, the vector  $\overline{\mathbf{\Delta}}_\tau$  consists of entries  $1/\Delta_{ws}$  and  $b_\tau = b/\tau^\gamma$  is defined with respect to the parameters  $b > 0$  and  $\gamma > 0$ . These parameters and the Bernoulli distribution are usually chosen because they satisfy certain conditions (Spall 1992) that ensure the algorithm to be able to converge to a minimum, asymptotically. To ensure compatibility with SUMO that only accepts integer inputs, we keep the perturbation parameter fixed  $b_\tau = 1$  throughout all iterations, which is equivalent to perturbing each element in the demand vector by  $\pm 1$  trip.

#### 4.1.3. Determining Step Lengths

The step length in the descent direction can be chosen appropriately by solving for  $\eta_\tau$  a line search problem in each iteration  $\tau$ :

$$\eta_\tau = \arg \min_{\alpha \geq 0} F(\mathbf{x}_\tau + \alpha \cdot \mathbf{d}_\tau). \quad (15)$$

The goal of the line search is to find the step rate  $\alpha$  that minimizes the OF in the descent direction given by  $\mathbf{d}_\tau$ . To solve this sub-problem we use a multiple linear regression approach where a 2nd-degree polynomial is fitted to sample points evaluated in the descent direction. More specifically, we first define a single-variable function of the step rate  $\alpha$ , as follows:

$$y(\alpha) = F(\mathbf{x}_\tau + \alpha \cdot \mathbf{d}_\tau). \quad (16)$$

Then, we determine:

- (1) An upper bound  $\ell$  on  $\alpha$  imposed by the generation constraints in Eq. (7). An algorithmic sketch to carry out this task is given in Algorithm 1.
- (2) A number of equally spaced points  $\alpha_1, \dots, \alpha_n$ ,  $n \geq 2$  to evaluate in the interval  $[\ell/n, \ell]$ .

Given these parameters we can obtain sample points with corresponding responses:

$$(\alpha_0, y(\alpha_0)), \dots, (\alpha_n, y(\alpha_n)).$$

The point  $(\alpha_0 = 0, y(\alpha_0) = F(\mathbf{x}_\tau))$  is already known. In this case, by specifying two additional sample points we are able to uniquely fit a 2nd-degree polynomial. More precisely, an estimator for the function  $y(\alpha)$  in Eq. (16) is:

$$\hat{y}(\alpha) = \beta_0 + \beta_1 \cdot \alpha + \beta_2 \cdot \alpha^2 \quad (17)$$

---

**Algorithm 1:** Procedure for determining the upper bound on the step rate.

---

```

1 Procedure DetermineMaxStepRate( $\mathbf{x}_\tau, \mathbf{x}_l, \mathbf{x}_u, \mathbf{d}_\tau, \ell_\Delta = 10^8, \epsilon_1 = 10^{-8}$ )
2    $\ell \leftarrow 0$ 
3   while true do
4      $\ell \leftarrow \ell + \ell_\Delta$ 
5      $\check{\mathbf{x}} \leftarrow \mathbf{x}_\tau + \ell \cdot \mathbf{d}_\tau$ 
6     // Apply upper and lower bounds
7      $\check{\mathbf{x}} \leftarrow \text{maximum}(\mathbf{x}_l, \check{\mathbf{x}})$ 
8      $\check{\mathbf{x}} \leftarrow \text{minimum}(\mathbf{x}_u, \check{\mathbf{x}})$ 
9     // Check if generation constraints are violated or whether
10    // all upper or lower bound constraints are binding
11    if  $\sum_{\substack{j \in D, w=(i,j) \in W \\ s \in S}} \check{x}_{ws} > o_i \forall i \in O \vee$ 
12     $(\check{x}_{ws} == x_{ws}^{\text{Lower}} \vee \check{x}_{ws} == x_{ws}^{\text{Upper}}) \forall w \in W, s \in S$  then
13      // Backtrack and try again with a smaller step rate
14       $\ell \leftarrow \ell - \ell_\Delta$ 
15       $\ell_\Delta \leftarrow \ell_\Delta / 2$ 
16      if  $\ell_\Delta < \epsilon_1$  then
17        | break
18    // Return the largest positive step rate that does not violate any
19    // constraints
20  return  $\ell$ 

```

---

where the coefficients  $\beta_0, \dots, \beta_2$  are determined by *least squares*, given the sample points and corresponding responses. The value  $\alpha_{\min}$  that minimizes this polynomial is given by  $\alpha_{\min} = \frac{-\beta_1}{2\beta_2}$ , if  $\beta_2 > 0$ . Otherwise,  $\alpha_{\min}$  is taken to be the step rate that produced the sample point with the smallest response.

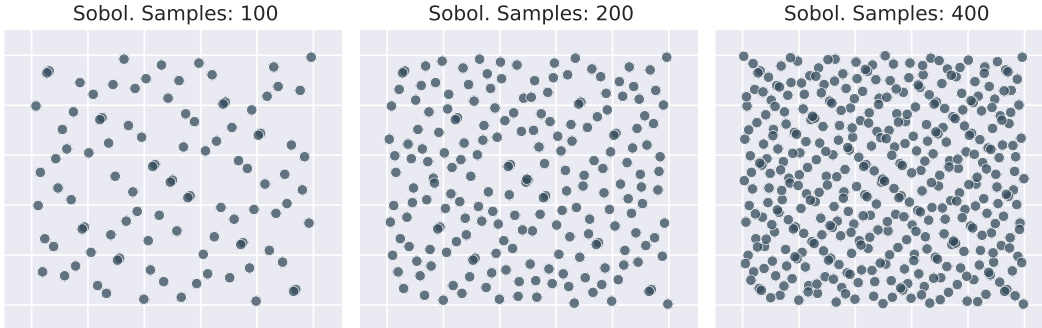
#### 4.1.4. Computational Complexity of the Gradient-based Approaches

If we look at the computational cost associated with the AM-based approach, we notice that two OF evaluations and thus DTA problems are solved in each iteration. First, a single DTA is needed to obtain a (vectorized) AM  $\mathbf{p}_\tau$  such that the descent direction and two additional points (using the relation in Eq. (9)) in the descent direction can be evaluated, and a 2nd-degree polynomial can be fitted. Using the 2nd-degree polynomial, the best step length in the descent direction can be determined. Then, one additional DTA is solved to evaluate the new point that has been found by using the best step length that minimizes the OF in the descent direction.

On the other hand, the AM-free approach needs to solve up to four DTA problems in each iteration, i.e., one DTA is required to estimate the gradient, and two additional DTAs are needed for determining the best step length in the descent direction. As with the AM-based approach, an additional DTA is then necessary to evaluate a new estimate found.

## 4.2. Machine Learning Approaches

The DTA performed by SUMO can be regarded as a black-box function that maps input demands to network-specific quantities, such as arc counts, speeds, densities, etc. As mentioned in Section 3, we will focus on the arc counts. In this case, the black-box function representing the DTA performed by SUMO can be described by Eq. (5), a vector-valued multivariate function that describes the input-output relationship of



**Figure 1.** Example of a sampling strategy in the plane that uses a Sobol low-discrepancy sequence.

the DTA performed by SUMO with respect to the output quantity we are interested in modeling. The ML approach’s main idea is to learn a model of this input-output relationship.

A dataset is needed to learn a model of the input-output relationship. This dataset can be obtained through a *sampling strategy*, which is a technique for deciding how many and which points should be included in a dataset provided to a *learning algorithm*. Using the dataset, the learning algorithm is trained in a supervised fashion, which results in a model  $\hat{\Gamma}$  for the relationship in Eq. (5). Subsequently, the DODE problem can be solved using the obtained model as a surrogate for DTA in the OF.

Overall, the ML approach put forth here consists of three primary components: a sampling strategy, the application of a ML algorithm and a final application of an optimization algorithm to obtain an approximate minimum of the DODE problem that is then to be evaluated and returned as a final estimate. These steps are described in more detail in the following.

#### 4.2.1. Sampling Strategy

The sampling strategy should preferably be chosen such that the amount of information gained from a limited number of sample points is maximal. Sampling more points than necessary implies additional computational resources and time. In other words, for the dataset we want to obtain a set of points that are sampled evenly in the search space. A way to accomplish this is by using *optimal experimental designs*, which among others include: Box-Behnken, central composite, factorial designs, and latin hypercube sampling, to name a few. These sampling strategies are specialized because they are tailored for fitting specific parametric models (e.g., linear or multiple linear models). Moreover, they dictate the exact number of samples needed to end up with a set of sample points that covers the search space well. Consequently, these strategies are well-suited for fitting models when prior knowledge of the model’s structure is available, and the sampling budget allows for the number of samples required by the sampling strategy. If this is not the case, an alternative is to use a strategy that samples points according to a *low-discrepancy sequence* (also referred to as a quasi-random sequence). Examples of low-discrepancy sequences include Halton, Hammersley, Niederreiter, and Sobol (see Pardalos, Zhigljavsky, and Žilinskas (2016) for more theoretical details on low-discrepancy sequences).

An advantage of the strategies that generate sample points according to low-

discrepancy sequences is that they can evenly cover the search space, regardless of the number of samples. It can also be possible to add additional samples later to fill the search space progressively.

Ultimately, the chosen sampling strategy depends on (i) the dimensionality of the problem and (ii) the simulation budget, which imposes an upper bound on the number of points to evaluate, i.e., the number of DTAs to perform.

In our experiments (see Section 6), we will use a dataset with points sampled according to the numbers in a Sobol sequence (an example is shown in Figure 1)).

Formally, let  $\mathcal{D}_\Gamma = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$  denote the dataset where  $\mathbf{x}_i \in \mathbb{R}_+^{m \cdot n_s}$  is an element of a Sobol sequence and  $\mathbf{y}_i = \Gamma(\mathbf{x}_i) \in \mathbb{R}_+^{n_q \cdot n_s}$  for all  $i \in \{1, \dots, n\}$ . Recall that  $\Gamma$  represents the function applied by the SUMO simulator. The learning task is to determine  $\hat{\Gamma}$  such that it approximates well  $\Gamma$  on new data points.

#### 4.2.2. Machine Learning Algorithms

A wide selection of off-the-shelf implementations of general-purpose ML algorithms can be used to find  $\hat{\Gamma}$ . The ML algorithms we are interested in studying can be categorized as nonparametric models. In contrast to parametric models, nonparametric models are more flexible as no prior assumptions are made on the structure of the model that is to be trained.

It could be possible to learn the input-output relationship between the demand and the scalar value of the second OF term in Eq. (4), that is:

$$f^{(2)} : \mathbb{R}_+^{n_q \cdot n_s} \rightarrow \mathbb{R}_+, \quad (18)$$

This term of the OF measures the discrepancy between the observed arc counts and the current estimate of arc counts, which depends indeed on the DTA. On the contrary, the other term in Eq. (4),  $f^{(1)}$ , measures the discrepancy between the estimated demands and the seed demands and does not require performing a DTA.

However, preliminary experiments showed that a better approach is trying to learn the input-output relationship of a vector-valued multivariate function as in Eq. (5) and predict the vector of arc counts. Note that this approach has the advantage of being more robust to a change in the number of sensors in the network (e.g., due to failure), since in that case the corresponding predicted arc counts can be ignored in the calculation of  $f^{(2)}$ . Learning a single model to predict  $p$  target variables can be done by inherently multi-target ML algorithms, which are able to exploit inter-target correlations. The inherently multi-target ML algorithms that we will make use of are *Feed-Forward Networks (FNNs)* and *k-Nearest Neighbors (k-NN)*. The algorithms are briefly described in the following and in more detail in Appendix B.

A *Feed-Forward Network (FNN)* is a type of neural network consisting of *units* arranged in layers: An input layer, a number of hidden layers and an output layer. A FNN architecture is thus mainly defined by the number of layers (the depth of the network  $m_d \in \mathbb{N}$  not counting the input layer) and the number of units in each layer (the width of a layer  $l_i \in \mathbb{N}$ , with  $i \in \{1, \dots, m_d\}$ ), along with the *activation functions* used in the different layers. In the network architectures that we consider a *ReLU* ( $\phi(\cdot) = \max(0, \cdot)$ ) *activation function* is used in the units situated in the hidden layers and a linear activation function is used in the output layer, since the task of learning  $\Gamma$  is a regression task. Finally, to reduce overfitting and improve the out-of-sample predictive performance a regularization parameter  $\lambda \in \mathbb{R}_+$  is also set. An appropriate value for this parameter is usually chosen using out-of-sample *Cross-Validation (CV)*



techniques.

$k$ -NN is one of the simplest nonparametric models used for regression. To predict the value  $\hat{y}$  of a new unseen point  $\mathbf{x}$ ,  $k$  needs to be specified. Its value specifies the number of points from a training set  $\mathcal{D}$  closest to  $\mathbf{x}$  that should be used to calculate the predicted value  $\hat{y}$ . Unlike the FNN and most other ML algorithms, no model must be determined, rather the training data are kept and continuously used.

It is good and common practice for supervised learning tasks to perform *model selection and validation* to estimate the out-of-sample predictive performance and obtain a model that ultimately has good *prediction accuracy* and generalizes well to unseen data. To perform model selection and validation, we will use the MSE as a performance metric to evaluate the performance of a trained model.

To select an appropriate FNN or  $k$ -NN model and estimate its performance on unseen data, we will perform *hyperparameter tuning* through an exhaustive grid search and use  $k$ -fold CV (note that this  $k$  is different from  $k$  in  $k$ -NN) on a dataset. For a hyperparameter configuration,  $k$ -fold CV consists of the following steps: (i) Divide the dataset into  $k$  non-overlapping groups of approximately equal size, then (ii) save a single fold as a validation dataset and use an ML algorithm to train a model on the remaining  $k - 1$  folds. Lastly, (iii) assess the performance of the trained model by calculating the MSE on the held-out fold. Steps (i) and (ii) are repeated  $k$  times where a new fold is treated as the validation dataset. Through this procedure, we obtain error estimates  $MSE_1, MSE_2, \dots, MSE_k$  that can be used to calculate the overall  $k$ -fold CV score:

$$CV_k = \frac{1}{k} \sum_{i=1}^k MSE_i \quad (19)$$

The  $CV_k$  score estimates the error we can expect when using a trained model to predict new values given unseen data. The model with the best score is chosen in a model selection context. The value of  $k$  is commonly set to 5 or 10, usually due to computational considerations (James et al. 2013).

#### 4.2.3. Optimization

We can then solve the outer-level optimization problem in Eq. (2) with  $\hat{\Gamma}$  to model the relationship in Eq. (5). So instead of obtaining arc counts by performing a time-consuming DTA with SUMO, we get arc counts by querying the ML model  $\hat{\Gamma}$ . Since we use nonparametric ML algorithms, the OF is not explicitly available. Hence, we need to resort to derivative-free methods. However, now evaluating the OF has become fast, and we can use algorithms that are less parsimonious in requiring this.

We can thus apply an off-the-shelf global optimization algorithm called the *basinhopping* algorithm, described in Wales and Doye (1997) and implemented by the SciPy Python library (Virtanen et al. 2020). The basinhopping algorithm is similar to the well-known *simulated annealing* optimization algorithm in the sense that it consists of two main components that are applied iteratively: (i) A random perturbation of a current point and (ii) a criterion for accepting or rejecting the perturbed point based on its OF value.

The basinhopping algorithm extends the idea of the simulated annealing algorithm by applying a local optimization algorithm to the perturbed point to find a local minimum of the OF before deciding to accept or reject the point that resulted in the local minimum. This process is repeated until a maximum number of iterations has

been reached. The local optimization algorithm used in the basinhopping algorithm as a subprocedure is, in this work, chosen to be a Sequential Least-Squares Programming (SLSQP) algorithm also implemented by the SciPy library. This choice is because this algorithm can handle not only the upper and lower bound constraints in Eq. (3) but also the inequality constraints in Eq. (7).

Ultimately, the basinhopping algorithm is applied to find an approximate minimum of the DODE problem. SUMO will then evaluate the final demand estimate as a final result.

## 5. Experimental Setup

We decided to test and compare the methods described in a controlled environment by synthetically generating an artificial traffic network. To generate this test data, three primary components are needed: (i) a traffic network where possible OD pairs  $W$  have been identified, (ii) a discretization of the analysis period  $T$ , and (iii) a *ground-truth* vector of demands based on (i) and (ii). The specific parameters used in each of the experiments are described in Section 6. Here, we focus on details of components (i)-(iii).

The network can be defined as an *irregular grid network*. The network was generated to contain features that can be encountered in a real traffic network. More specifically, some paths in the network between origins and destinations are shorter than others in terms of free-flow travel time. The network used is depicted in Figure 2(a) and described in more detail in the following.

The network has 48 directed arcs and 16 nodes, either representing intersections or origins and destinations. More precisely, 12 of the nodes represent origins and destinations, which means that  $O = D$  and  $|O| = |D| = 12$ , resulting in 132 OD pairs. Furthermore, the arcs in the network were prescribed a speed limit of 50 kilometer/hour and assigned a default length of 1250 meters. The coordinates of the nodes in the network are then perturbed by a uniform random number  $\mathcal{U}(0, 1250)$ , resulting in arcs between nodes with different lengths, yielding the irregular grid.

We settled for a one-hour scenario with an analysis period  $T = [0, t_{\text{end}})$ , where  $t_{\text{end}} = 3600$  seconds = 60 minutes = 1 hour. This one-hour time period was discretized into 4 time intervals of equal duration, i.e., we have  $S = \{1, \dots, n_S = 4\}$ , where each of the indices in this set refers to a time interval of 900 seconds = 15 minutes duration. Given this information and a network with OD pairs  $W$ , a ground-truth vector of demands is constructed by sampling the number of trips defined for each variable according to a continuous uniform distribution with lower bound  $a$  and upper bound  $b$ :

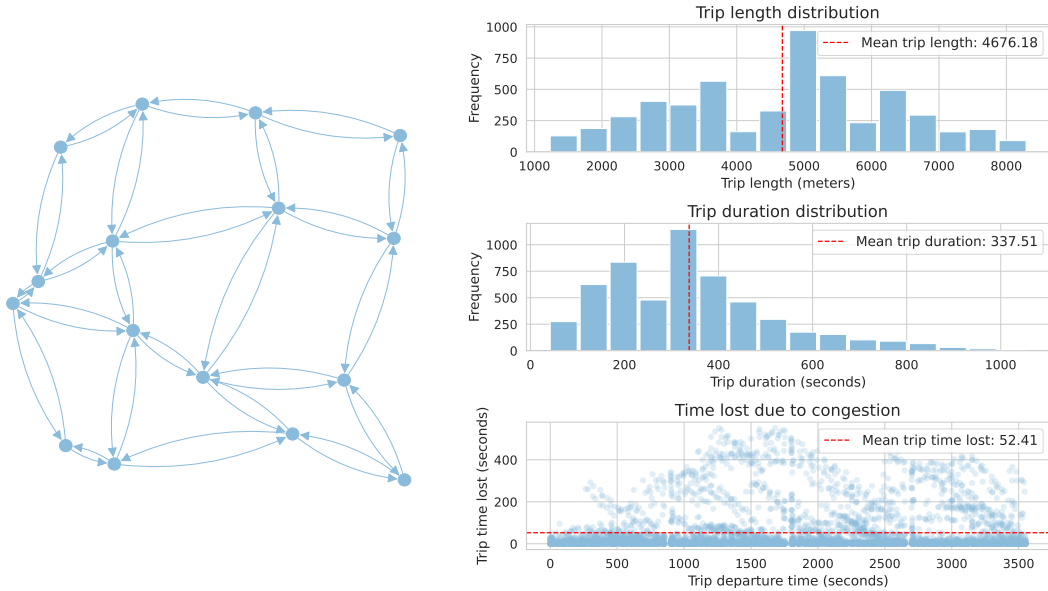
$$x_{ws}^{\text{True}} \sim U(a, b), \forall w \in W, s \in S. \quad (20)$$

From the ground-truth vector of demands, trip productions  $o_i$  for  $i = 1, \dots, |O|$  that enter into the generation constraints (in Eq. (7)) were obtained in addition to upper and lower bounds on the estimated demands:

$$\mathbf{x}_l = \mathbf{0} \text{ and } \mathbf{x}_u = 1.5 \cdot \max(\mathbf{x}) \cdot \mathbf{1}, \quad (21)$$

where  $\mathbf{0}$  is an  $m \cdot n_S$  vector of zeros and  $\mathbf{1}$  is an  $m \cdot n_S$  vector of ones.

Finally, the ground-truth vector of demands was given to SUMO as input and the



(a) The network used in the case study. All the nodes on the perimeter of the network can be identified as origins and destinations. (b) Characteristics of the synthetically generated test data using the artificial network.

**Figure 2.** The network used in the case study along with the characteristics of the synthetically generated test data.

iterative DTA procedure implemented by SUMO was run for 15 iterations yielding the vector  $\hat{c}$  of observed arc counts.

For the seed demand vector  $\tilde{x} \in \mathbb{R}^{m \cdot ns}$  we consider two scenarios:

**Prior low-demand vector (LD):** This scenario simulates the case where lower demand estimates are available from a previous study:

$$\tilde{x}_{ws} = x_{ws}^{\text{True}} \cdot (0.7 + 0.3 \cdot u_{ws}), \quad u_{ws} \sim U(0, 1), \quad \forall w \in W, \forall s \in S. \quad (22)$$

**Prior high-demand vector (HD):** This scenario simulates the case where higher demands are available from a previous study:

$$\tilde{x}_{ws} = x_{ws}^{\text{True}} \cdot (0.9 + 0.3 \cdot u_{ws}), \quad u_{ws} \sim U(0, 1), \quad \forall w \in W, \forall s \in S. \quad (23)$$

These two scenarios were constructed based on the guidelines provided in [Antoniou et al. \(2016\)](#) and the computational experiments performed in [Cipriani et al. \(2011\)](#); [Masip et al. \(2018\)](#).

The ground-truth vector of demands was generated according to Eq. (20), using lower bound  $a = 1$  and upper bound  $b = 20$ . The value for the upper bound  $b$  depends on the number of OD pairs and the size of the network, i.e., the network supply, which essentially determines how much traffic the network can handle at a given point in time. In this case, the appropriate upper bound was found experimentally by sampling different ground-truth vectors of demand for increasing values of  $b$ . A DTA was performed by SUMO for each of the generated ground-truth vectors, and the resulting output from the DTA was assessed by checking the level of congestion present in the network. Finally, the ground-truth vector of demands with corresponding

upper bound  $b$  that generated a scenario with light congestion was chosen as the basis for the case study using this network.

The scenario that was deemed the most appropriate was determined based on assessing the characteristics of the three plots in Figure 2(b). In the first plot, we observe that the average trip length is 4.6 kilometers and, in the next plot, that the average travel time for a trip is 338 seconds. Finally, keeping these two attributes in mind, we observe in the last plot that an average delay of 52 seconds ( $52 \text{ seconds} / 338 \text{ seconds} = 0.15\%$ ) of the total travel time is due to congestion and driving below the possible free-flow speed. Furthermore, some unfortunate vehicles experience a delay far more significant than the average of 52 seconds, while on the other hand, most vehicles experience only little to no travel time delays. Therefore, based on these scenario attributes, we classify the scenario as a scenario with light congestion.

For the results we present in the following sections, it is assumed that sensors cover 100% of the network. A good overview of the problem of determining appropriate locations to place sensors in a traffic network, along with some recent developments on this problem, is given in Viti et al. (2014); Hu and Liou (2014); Salari et al. (2019). In addition to this, the arc counts obtained by the sensors are assumed to be error-free. This is, of course, not the case in a real traffic network, but it simplifies the analysis and comparison of the different solution approaches. Lastly, we mention that all the experiments were executed using an initial feasible solution consisting of all ones, i.e.,  $\mathbf{x}_{\tau=0} = \mathbf{1}$ . This initial guess was chosen so to not confine the search for a minimum to the vicinity of the seed demands.

To compare the performance of the different approaches, we report the Root Mean Squared Error (RMSE). If we consider place-holders  $\mathbf{y}, \hat{\mathbf{y}} \in \mathbb{R}^n$ , where  $\mathbf{y}$  is a vector of observed values and  $\hat{\mathbf{y}}$  is a vector of estimated values, then the RMSE value is defined as follows:

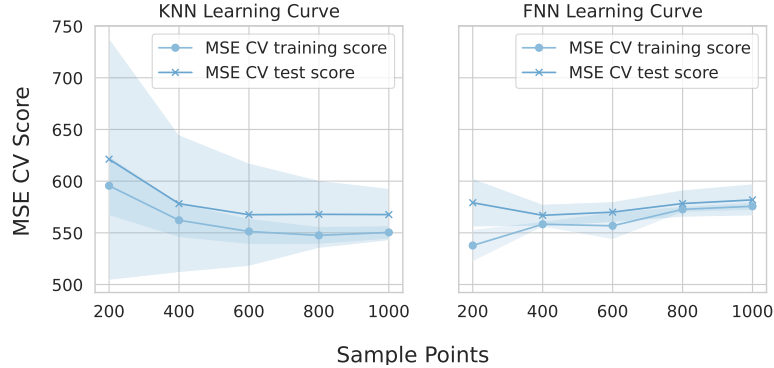
$$\text{RMSE}(\mathbf{y}, \hat{\mathbf{y}}) = \left( \frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2 \right)^{\frac{1}{2}}. \quad (24)$$

This error measure is easily interpretable because the error is given in the same units as the data. Furthermore, the RMSE is reported separately for each of the different quantities entering the optimization problem, i.e., we report the RMSE statistic for the arc counts and the demands. Finally, we remark that for the demands, the RMSE is calculated based on the ground-truth values and not on the seed demand vectors that enter the optimization problem.

We note that the size of the network and the number of OD pairs, arcs equipped with sensors and time intervals of the dataset described are similar to those used in other algorithm comparison studies (Cipriani et al. 2011; Carrese et al. 2017; Nigro et al. 2018), while larger sizes and numbers are used in Toledo and Kolehkina (2013); Antoniou et al. (2016); Djukic et al. (2017); Masip et al. (2018).

## 6. Experimental Results

We implemented all methods in Python with external calls to the SUMO simulator. The primary Python libraries used include Pandas (McKinney 2010) for data organization purposes, NumPy (Harris et al. 2020) for its numerical methods for fitting polynomials, along with SciPy (Virtanen et al. 2020) for its off-the-shelf global opti-



**Figure 3.** The learning curves and CV results associated with the training of the  $k$ -NN and FNN model. The curves show the mean CV and training scores, while the shaded bands show the standard deviation. The datapoints in the curves are related to the best mean CV score obtained for different sample sizes and over all tested hyperparameter configurations.

mization algorithms and scikit-learn’s (Pedregosa et al. 2011) implementation of FNN and  $k$ -NN used in the ML approach. A snapshot of the repository (Andersen 2022a) containing the codebase and the data supporting this study’s findings are archived and openly available in Zenodo (Andersen 2022b). All experiments described in this section were performed on a machine with an AMD Ryzen 7 1800X eight-core processor and 32 GB of RAM, running Manjaro Linux.

### 6.1. Computational Budget

We studied the ability of the FNN and  $k$ -NN ML algorithms to learn the input-output relationship using a different number of sample points and hyperparameter configurations while the CV scores were monitored. The resulting *learning curves* are displayed in Figure 3. We observe that by increasing the size of the training set, the CV scores improve but only slightly.

As we want a model that performs as well as possible on unseen data (the test scores) and is parsimonious in the usage of SUMO to perform DTAs, we settled to use 200 points for the dataset to train the ML approaches. This means that we run SUMO for about 200 DTAs, which took 3912.92 sec = 65.21 min on our computational environment. We note that an additional DTA (the 201st DTA) will be needed by the ML approaches to evaluate a final estimate. For the sake of fairness, we thus set a soft upper bound on the number of allowed OF evaluations to 201 for all other approaches<sup>1</sup>.

### 6.2. Parameter Tuning

The gradient-based approaches, i.e., the AM-based and AM-free approach, do not need parameter tuning before tackling the DODE problem. Indeed, the only parameter, the step length, is determined automatically by the solution of the line search in Eq. (15). Thus, these algorithms are straightforward to apply, and no specific consideration has to be made concerning their parameters.

For the ML approaches, the FNN and the  $k$ -NN model, the hyperparameters were

<sup>1</sup>Due to the inherent nature of some of the applied approaches, a few more OF evaluations and thus DTAs might have occurred in an iteration and the total number of OF evaluations might not sum up to 201 exactly.

**Table 1.** An overview and summary of the results obtained in the different experiments E1-E12. Here, we define M1: AM-based approach, M2: AM-free approach, M3: FNN predicting  $\Gamma$ , M4:  $k$ -NN predicting  $\Gamma$ . For methods M3-M4 only the running time of the optimization step is reported.

Method	Experiment	Seed matrix	OF evaluations	Running time (sec)	RMSE( $\mathbf{x}$ , $\mathbf{x}^{\text{True}}$ )	RMSE( $\mathbf{c}(\mathbf{x})$ , $\hat{\mathbf{c}}$ )
M1	E1	None	201	2836.4469	1.3413	9.4626
	E2	LD	201	2845.5479	1.3413	9.4626
	E3	HD	201	2770.9962	1.2516	9.6358
M2	E4	None	203	2552.5320	9.0188	19.4008
	E5	LD	201	2366.2931	2.0342	13.5121
	E6	HD	202	2475.8943	6.7992	16.1219
M3	E7	None	201	110.9053	9.6494	20.8409
	E8	LD	201	116.7221	9.6338	20.4193
	E9	HD	201	107.7270	9.6493	20.8409
M4	E10	None	201	103.1716	9.6728	21.1067
	E11	LD	201	109.7821	9.6338	20.4193
	E12	HD	201	102.5330	9.6339	20.4193

fine-tuned using a grid search coupled with 5-fold CV using the 200 generated sample points that act as re-usable historical simulation data.

The grid search was performed on the values reported in Table 2. The best performance was achieved by the following settings:

- The FNN model with hidden layers containing units  $(0.75 \cdot m \cdot n_S, 0.50 \cdot m \cdot n_S, 0.50 \cdot m \cdot n_S) = (l_1 = 396, l_2 = 264, l_3 = 264)$  along with regularization parameter  $\lambda = 0.01$ . This trained model is used in method M3 in Table 1. This model achieved a 5-fold CV score of  $CV_5 = 579.1282$ .
- The  $k$ -NN model using  $k = 43$  neighbours. This trained model is used in method M4 in Table 1. This model achieved a 5-fold CV score of  $CV_5 = 621.2651$ .

For the subsequent optimization step using a trained ML model (FNN or  $k$ -NN) as a surrogate model for DTA, the basinhopping algorithm was used with default parameters. Also, the local SLSQP minimizer was used with default parameters. Preliminary experiments showed that 10 iterations of the basinhopping algorithm were enough to reach convergence and that improvements after that stage become rare.

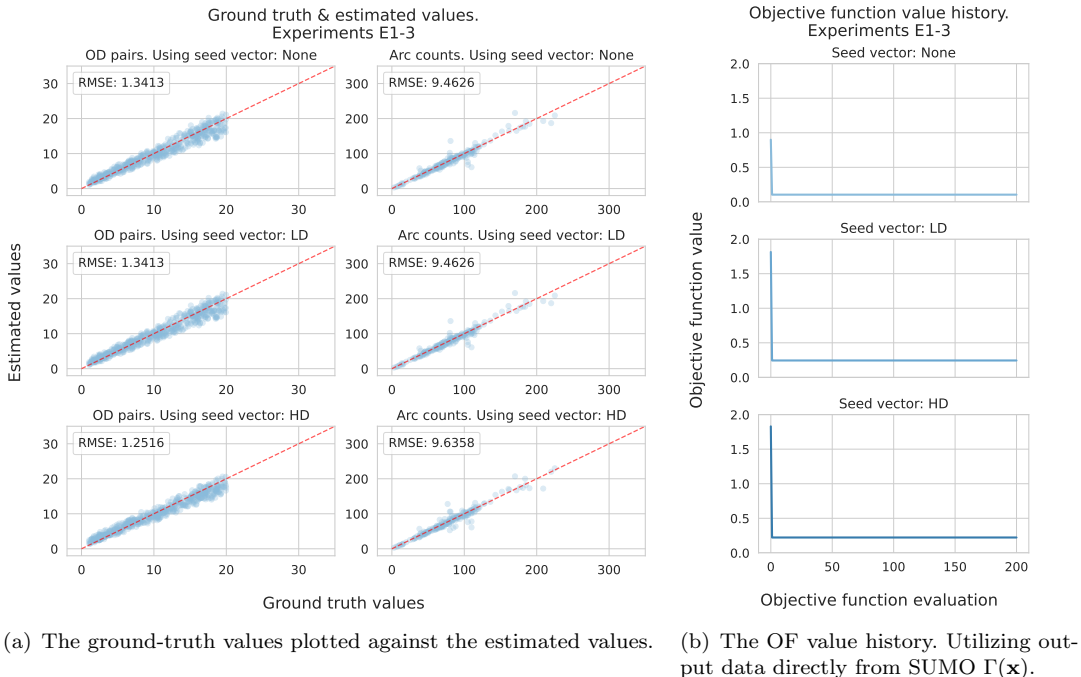
### 6.3. Quality of Estimates

In Table 1, we present an overview of all experiments conducted together with summary results, that we detail in the following.

#### 6.3.1. Gradient-based Approaches

The final results obtained from the gradient-based approaches are plotted in Figure 4(a) and Figure 5(a). In these figures, the estimated quantities  $\mathbf{x}$  and  $\mathbf{c}(\mathbf{x})$  have been plotted against the ground-truth vector of demands  $\mathbf{x}^{\text{True}}$  and the corresponding vector of observed arc counts  $\mathbf{c}(\mathbf{x})$ , respectively. The corresponding plots that show the OF value history of the two algorithms are shown in Figure 4(b) and Figure 5(b).

If we compare the results in Figure 4(a) and Figure 5(a) between the two gradient-based approaches, we observe that the AM-based approach outperforms the AM-free approach. In fact, the AM-based approach is able to consistently overcome local minima and reach a good demand estimate. This can be seen in Figure 4(a) where the



**Figure 4.** The final results from experiments E1-E3, that used the gradient and AM-based approach.

estimated demands and the ground-truth vector of demands align well, and so do also the estimated arc counts and the observed arc counts.

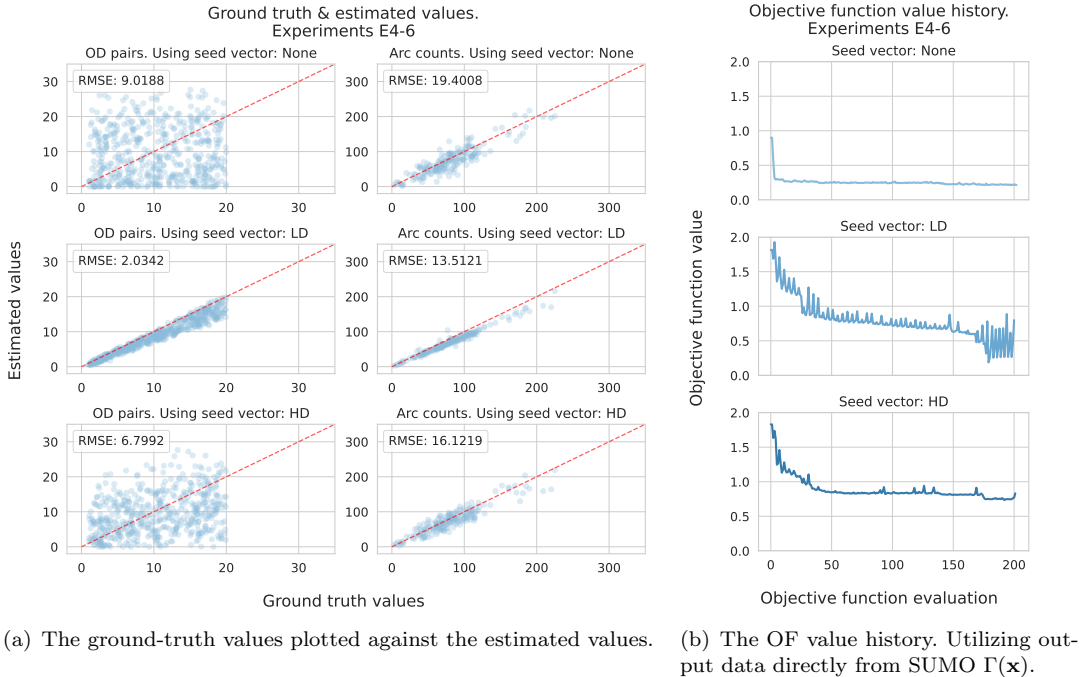
If we look at the progress of the two algorithms in Figure 4(b) and Figure 5(b), we see that the AM-based approach can find a good estimate within only a single OF evaluation, while the AM-free approach uses a much larger number of evaluations to arrive at good estimates.

If we return to Table 1 and the results relating to methods M1 and M2’s computational time, we observe that the AM-free (M2) appears to be faster than the AM-based (M1) approach. This is the case, even though two OF evaluations and thus DTA computations are needed in each iteration of M1, while M2 needs up to four. The prolonged computational time can be attributed to slow gradient calculations due to AM data organization, which takes time in Python. As the network and scenario are scaled, we expect the case to be inverted, i.e., data organization will take less time than performing DTA’s.

### 6.3.2. ML Approaches

The final estimate reached by the  $k$ -NN approach is shown in Figure 6 and the one reached by the FNN approach in Figure 7. The figures also report the development of solution quality over the 10 iterations of the basinhopping algorithm which typically yield around 6000 OF evaluations using the surrogate model  $\tilde{\Gamma}$ .

Across the different experiments with the ML approaches, we observe that the two ML models achieve very similar results. Under both models, the estimated and observed arc counts align well but the ground-truth demands and the estimated demands do not. In other terms, these approaches can find demand estimates that reproduce the arc counts well but are in no way similar to the ground-truth demands.



**Figure 5.** The final results from experiments E4-E6, that used the gradient and AM-free approach.

In the optimization step of the ML approaches, we observe a general lack of progress. In fact, an examination of the quality of the data points against the final point after the optimization step show that the ML approaches seem able to improve only slightly the estimates already available initially.

Furthermore, if the OF values pertaining to the first and the second term of the OF is inspected, then it appears that the first term (in Eq. (6)) relating to the demands is not worked on as much as the second term that takes into account the arc counts. This behavior helps explain why the ground-truth demands and the estimated demands do not align well even when a seed vector is provided to guide the search. Another reason for the overall poor performance can also be attributed to the goodness-of-fit of the trained ML models. If the trained models do not sufficiently capture the complex input-output relationship of a DTA performed by SUMO, then we might only be able to explore or reach local minima of the DODE problem in the subsequent optimization step that are not actual minima. A hint that this might be the case is the high CV scores reported in Figure 3.

**Table 2.** The ML model hyperparameters chosen for hyperparameter tuning. Preliminary experiments were performed to narrow down the domains of the hyperparameters.

Machine learning alg.	Hyperparameter name	Domain
FNN	Architecture	$\{(0.75 \cdot m \cdot n_S, 0.50 \cdot m \cdot n_S, 0.50 \cdot m \cdot n_S), (0.75 \cdot m \cdot n_S, 0.25 \cdot m \cdot n_S, 0.25 \cdot m \cdot n_S), (0.75 \cdot m \cdot n_S, 0.125 \cdot m \cdot n_S, 0.125 \cdot m \cdot n_S)\}$
	Regularization $\lambda$	$\{0.0001, 0.001, 0.01, 0.1\}$
$k$ -NN	Neighbours $k$	$\{\lfloor 2^z \rfloor : z = 2 + i \cdot (6^{-2})/14, \forall i = 0, \dots, 14\}$



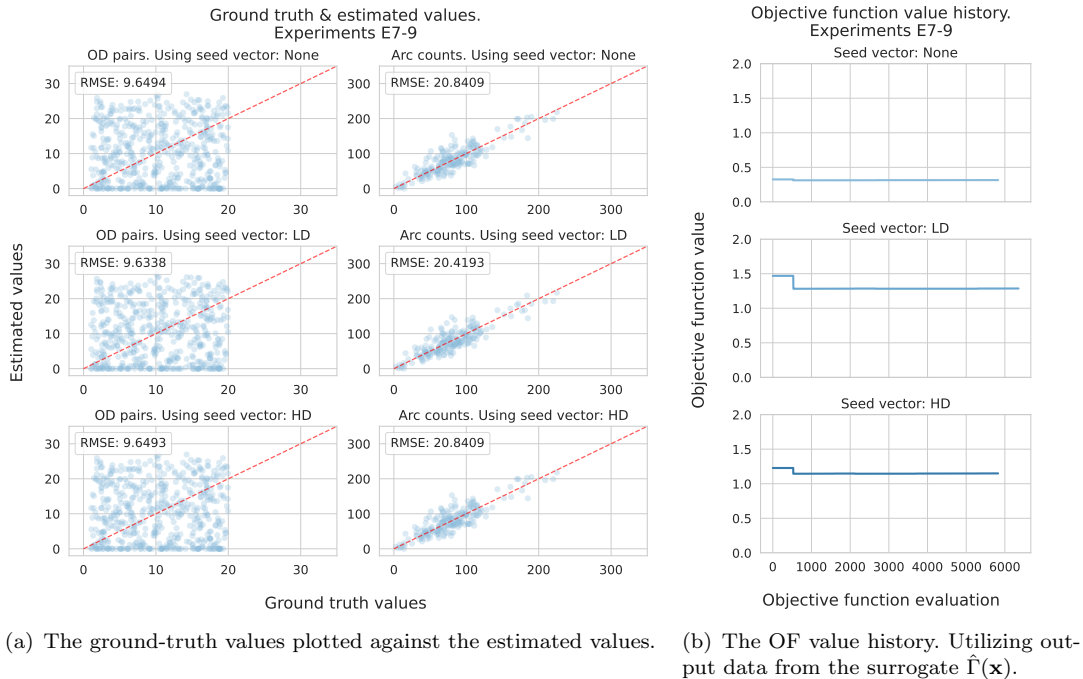


Figure 6. The final results from experiments E7-E9, where the  $k$ -NN model predict  $\Gamma$ .

#### 6.4. Scaling

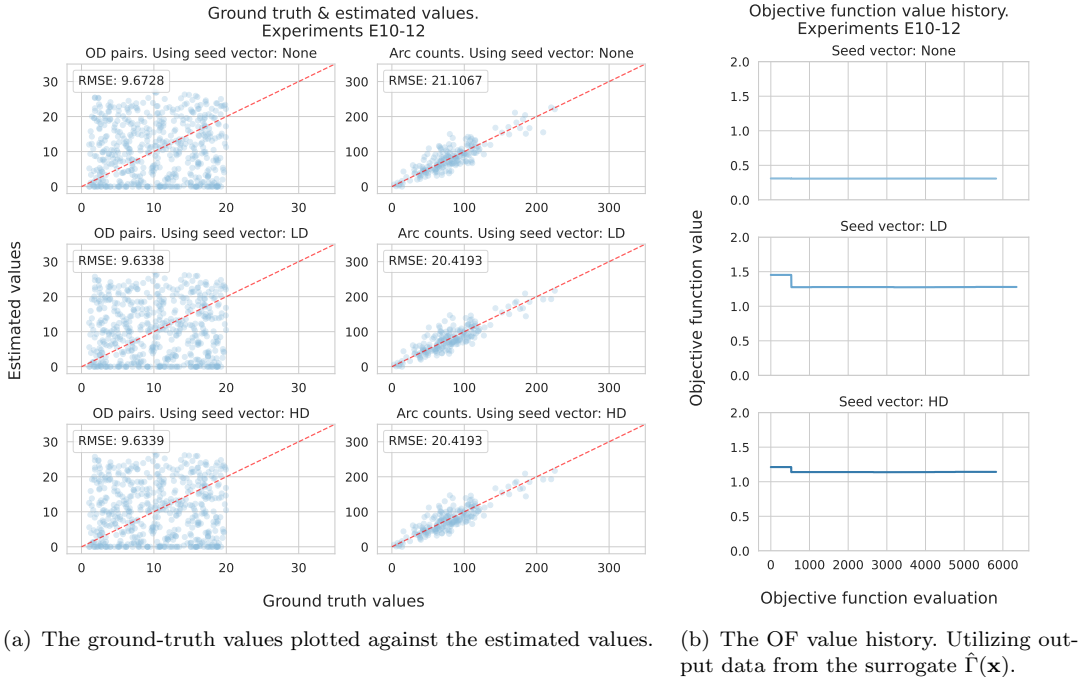
Under the assumption that the training set of data is available initially in the form of historical data, then performing the optimization step on the ML models took on average 108.47 sec (average running time of M3-M4 in Table 1). In contrast, the average time for the gradient-based approaches using SUMO directly for DTA was 2641.28 sec (average running time of M1-M2 in Table 1).

Further, depending on the network size, the ML approaches could scale better than the gradient-based approaches. Compared to the gradient-based approaches that perform many DTAs, starting from scratch in each new optimization task, the ML approaches utilize existing DTA data, meaning that fewer DTAs overall might be needed.

A factor affecting the running time of all approaches is the number of demand variables to be estimated. Increasing the number of variables produces a more complex estimation problem that results in many more DTAs having to be solved for an approach to find improving estimates. To cope with this growth, it is possible to manually adjust the number of variables, e.g., by merging OD pairs by combining areas into larger zones or by adjusting the number of sub-intervals of the analysis period in which the demand is estimated. Moreover, arcs equipped with sensors can be prioritized, and arcs deemed less critical can be filtered out, making it possible to scale to larger networks.

## 7. Conclusions and Future Work

The results presented in Table 1 and the plots in Figure 4, 5, 6, and 7 show that the AM-based approach (method M1) outperforms, in terms of quality of the estimates produced, the AM-free approach (method M2), as well as the ML approaches (methods



**Figure 7.** The final results from experiments E10-E12, where the FNN model predict  $\Gamma$ .

M3-M4). We also note that even if no seed vectors were supplied, the AM-based approach could find a reasonable estimate for both the ground-truth demands and the corresponding observed arc counts.

A further advantage of the AM-based approach is that it is easy to apply as no parameters will have to be set or tuned before using the approach. Moreover, only two new OF evaluations and thus DTA computations are needed in each iteration, while in comparison, the AM-free approach needs up to four. On the other hand, the disadvantage of the AM-based approach is that it might be harder to include different, more complex types of data into the optimization problem, as it can be harder to derive analytical expressions for the gradient of function terms that take into account more complex data types such as road queuing length, turning ratios, etc.

If we look at the results obtained with the ML approach in Table 1 and Figures 6 and 7, we can conclude that the estimated and observed counts align somewhat well, while the estimated demands and ground-truth demands do not align at all. This result seems to remain the same independently of which ML algorithm is used. Based on these results, the ML approach is not good if one seeks a demand estimate that is sufficiently close to a given seed vector. However, it can produce more diverse estimates than, e.g., the AM-based approach and may be useful for other purposes.

Our conclusion is that the gradient-based approaches work the best. However, the ML methodology does provide a way to utilize data generated through many DTAs, and it might be a viable option if certain changes are made. In this case, we suggest possible directions for future research to improve the ML approach:

- In the optimization step of the ML approach, investigate why only the second OF term, relating to the arcs counts, appears to have an effect. That is, determine why the first term pertaining to the demands is not being optimized and able to guide the search even when a seed vector is supplied.

- Incorporate additional constraints into the optimization step of the ML approach to further restrict the search space and possibly obtain better estimates.
- Use other types of information, e.g., about the complete paths used by users in the network instead of just the arc counts.
- Model a different part of the DODE problem and change the type of prediction being made (i.e., Eq. (18)) by using more sophisticated ML models. For example, it can be conjectured that by using Graph Neural Networks (GNN), it is possible to capture the cause and effect better of the demand on the resulting arc counts as the relationships and interdependencies that exists between the arcs and thus arc counts in the traffic network are modeled more explicitly.

## References

- Abrahamsson, Torgil. 1998. “Estimation of origin-destination matrices using traffic counts - a literature survey.” (Interim Report) International Institute for Applied Systems Analysis, Laxenburg, Austria, <http://pure.iiasa.ac.at/5627>.
- Andersen, Nicklas Sindlev. 2022a. “Dynamic Matrix Estimation in Traffic Networks (Codebase).” Published: 01-02-2022, <https://github.com/NicklasXYZ/sumotde>.
- Andersen, Nicklas Sindlev. 2022b. “Dynamic Origin-Destination Matrix Estimation in Traffic Networks (Codebase Snapshot).” Published: 31-01-2022, <https://doi.org/10.5281/zenodo.5910743>.
- Antoniou, Constantinos, Carlos Lima Azevedo, Lu Lu, Francisco Pereira, and Moshe Ben-Akiva. 2015. “W-SPSA in practice: Approximation of weight matrices and calibration of traffic simulation models.” *Transportation Research Part C: Emerging Technologies* 59: 129 – 146. Special Issue on International Symposium on Transportation and Traffic Theory.
- Antoniou, Constantinos, Ramachandran Balakrishna, Haris N. Koutsopoulos, and Moshe Ben-Akiva. 2009. “Off-line and on-line calibration of Dynamic Traffic Assignment Systems.” *IFAC Proceedings Volumes* 42 (15): 104 – 111. 12th IFAC Symposium on Control in Transportation Systems.
- Antoniou, Constantinos, Jaume Barceló, Martijn Breen, Manuel Bullejos, Jordi Casas, Ernesto Cipriani, Biagio Ciuffo, et al. 2016. “Towards a generic benchmarking platform for origin-destination flows estimation/updating algorithms: Design, demonstration and validation.” *Transportation Research Part C: Emerging Technologies* 66: 79 – 98. Advanced Network Traffic Management: From dynamic state estimation to traffic control.
- Ashok, K., and Moshe Ben-Akiva. 2000. “Alternative Approaches for Real-Time Estimation and Prediction of Time-Dependent Origin—Destination Flows.” *Transportation Science* 34 (1): 21–36.
- Audet, Charles, and Warren Hare. 2017. *Derivative-Free and Blackbox Optimization*. Springer International Publishing.
- Balakrishna, Ramachandran. 2006. “Off-line calibration of dynamic traffic assignment models.” (PhD thesis) Massachusetts Institute of Technology, Cambridge, MA, USA, <https://dspace.mit.edu/handle/1721.1/35120>.
- Bert, Emmanuel. 2009. “Dynamic Urban Origin-Destination Matrix Estimation Methodology.” (PhD thesis) École polytechnique, Paris, France, [https://infoscience.epfl.ch/record/135711/files/EPFL\\_TH4417.pdf](https://infoscience.epfl.ch/record/135711/files/EPFL_TH4417.pdf).
- Carrese, Stefano, Ernesto Cipriani, Livia Mannini, and Marialisa Nigro. 2017. “Dynamic demand estimation and prediction for traffic urban networks adopting new data sources.” *Transportation Research Part C: Emerging Technologies* 81: 83 – 98.
- Cipriani, Ernesto, Michael Florian, Michael Mahut, and Marialisa Nigro. 2011. “A gradient approximation approach for adjusting temporal origin-destination matrices.” *Transportation Research Part C: Emerging Technologies* 19 (2): 270 – 282. Emerging theories in traffic and transportation and methods for transportation planning and operations.
- Djukic, Tamara. 2014. “Dynamic OD Demand Estimation and Prediction for Dynamic Traffic Management.” (PhD thesis) Delt University of Technology, Delft, The Netherlands.
- Djukic, Tamara, David Masip, Martijn Breen, and Jordi Cascas. 2017. “Modified bi-level optimization framework for dynamic OD demand estimation in the congested networks.” *39th Australasian Transport Research Forum (ATRF 2017)* 1 – 15. <https://trid.trb.org/view/1596710>.
- Frederix, Rodric, Francesco Viti, and Chris M. J. Tampère. 2013. “Dynamic origin-destination estimation in congested networks: theoretical findings and implications in practice.” *Transportmetrica A: Transport Science* 9 (6): 494–513.
- Friesz, Terry L., and David Bernstein. 2016. *Foundation of Network Optimization and Games*. Springer International Publishing.
- Gawron, Christian. 1998. “Simulation-Based Traffic Assignment.” (PhD thesis) University of Cologne, Cologne, Germany, <https://sumo.dlr.de/pdf/GawronDiss.pdf>.

- German Aerospace Center. 2022. “Meso.” Accessed: 25-02-2022, <https://sumo.dlr.de/docs/Simulation/Meso.html>.
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. “Array programming with NumPy.” *Nature* 585 (7825): 357–362.
- Hu, Shou-Ren, and Han-Tsung Liou. 2014. “A generalized sensor location model for the estimation of network origin–destination matrices.” *Transportation Research Part C: Emerging Technologies* 40: 93 – 110.
- James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning: With Applications in R*. Springer-Verlag New York.
- Kattan, Lina, and Baher Abdulhai. 2006. “Noniterative Approach to Dynamic Traffic Origin–Destination Estimation with Parallel Evolutionary Algorithms.” *Transportation Research Record* 1964 (1): 201–210.
- Lindveld, Charles Dirk Rudolf. 2003. “Dynamic O-D matrix estimation: A behavioural approach.” (PhD thesis) Delt University of Technology, Delft, The Netherlands, <http://resolver.tudelft.nl/uuid:fbe10ed4-0c17-4d2f-a272-30ce17399174>.
- Lopez, Pablo Alvarez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. “Microscopic Traffic Simulation using SUMO.” *IEEE Intelligent Transportation Systems Conference (ITSC)* <https://elib.dlr.de/124092/>.
- Lu, Lu, Yan Xu, Constantinos Antoniou, and Moshe Ben-Akiva. 2015. “An enhanced SPSA algorithm for the calibration of Dynamic Traffic Assignment models.” *Transportation Research Part C: Emerging Technologies* 51: 149 – 166.
- Marzano, Vittorio, Andrea Papola, and Fulvio Simonelli. 2009. “Limits and perspectives of effective O-D matrix correction using traffic counts.” *Transportation Research Part C: Emerging Technologies* 17 (2): 120 – 132.
- Masip, David, Tamara Djukic, Martijn Breen, and Jordi Cascas. 2018. “Efficient OD matrix estimation based on metamodel for nonlinear assignment function.” *40th Australasian Transport Research Forum (ATRF)* 1 – 15. <https://trid.trb.org/view/1589089>.
- McKinney, Wes. 2010. “Data Structures for Statistical Computing in Python.” In *Proceedings of the 9th Python in Science Conference*, edited by Stéfan van der Walt and Jarrod Millman, 56 – 61.
- Nigro, Marialisa, Ernesto Cipriani, Chiara Colombaroni, Gaetano Fusco, and Andrea Gemma. 2018. “Dynamic O-D Demand Estimation: Application of SPSA AD-PI Method in Conjunction with Different Assignment Strategies.” *Journal of Advanced Transportation* 1–18.
- Omran, Reza. 2014. “Off-line Multi-Sensor Multi-Source Calibration of Dynamic Traffic Assignment: Simultaneous Demand-Supply Estimation based on Genetic Algorithms in a High-Performance Computer.” (PhD thesis) University of Calgary, Calgary, Canada.
- Omran, Reza, and Lina Kattan. 2012. “Demand and Supply Calibration of Dynamic Traffic Assignment Models: Past Efforts and Future Challenges.” *Transportation Research Record* 2283 (1): 100–112.
- Osorio, Carolina. 2019. “High-dimensional offline origin-destination (OD) demand calibration for stochastic traffic simulators of large-scale road networks.” *Transportation Research Part B: Methodological* 124: 18 – 43.
- Pardalos, Panos M., Anatoly Zhigljavsky, and Julius Žilinskas. 2016. *Advances in Stochastic and Deterministic Global Optimization*. Springer International Publishing, Switzerland.
- Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12: 2825–2830. <https://dl.acm.org/doi/10.5555/1953048.2078195>.
- Peeta, Srinivas, and Athanasios K. Ziliaskopoulos. 2001. “Foundations of Dynamic Traffic Assignment: The Past, the Present and the Future.” *Networks and Spatial Economics* 1 (3):

233–265.

- Salari, Mostafa, Lina Kattan, William H.K. Lam, H.P. Lo, and Mohammad Ansari Esfeh. 2019. “Optimization of traffic sensor location for complete link flow observability in traffic network considering sensor failure.” *Transportation Research Part B: Methodological* 121: 216 – 251.
- Shafiei, Sajjad, Meead Saberi, Ali Zockaie, and Majid Sarvi. 2017. “Sensitivity-Based Linear Approximation Method to Estimate Time-Dependent Origin–Destination Demand in Congested Networks.” *Transportation Research Record* 2669 (1): 72–79.
- Spall, James C. 1992. “Multivariate stochastic approximation using a simultaneous perturbation gradient approximation.” *IEEE Transactions on Automatic Control* 37 (3): 332–341.
- Spall, James C. 1998. “Implementation of the simultaneous perturbation algorithm for stochastic optimization.” *IEEE Transactions on Aerospace and Electronic Systems* 34 (3): 817–823.
- Toledo, Tomer, and Tanya Kolechkina. 2013. “Estimation of Dynamic Origin–Destination Matrices Using Linear Assignment Matrix Approximations.” *IEEE Transactions on Intelligent Transportation Systems* 14 (2): 618 – 626.
- Tsekeris, Theodore, Loukas Dimitriou, and Antony Stathopoulos. 2007. “Simultaneous Origin–Destination Matrix Estimation in Dynamic Traffic Networks with Evolutionary Computing.” *Applications of Evolutionary Computing* 668–677.
- Tympakianaki, Athina, Haris N. Koutsopoulos, and Erik Jenelius. 2015. “c-SPSA: Cluster-wise simultaneous perturbation stochastic approximation algorithm and its application to dynamic origin–destination matrix estimation.” *Transportation Research Part C: Emerging Technologies* 55: 231 – 245. Engineering and Applied Sciences Optimization (OPT-i) - Professor Matthew G. Karlaftis Memorial Issue.
- Vaze, Vikrant, Constantinos Antoniou, Yang Wen, and Moshe Ben-Akiva. 2009. “Calibration of Dynamic Traffic Assignment Models with Point-to-Point Traffic Surveillance.” *Transportation Research Record* 2090 (1): 1–9.
- Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al. 2020. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python.” *Nature Methods* 17: 261–272.
- Viti, Francesco, Marco Rinaldi, Francesco Corman, and Chris M. J. Tampère. 2014. “Assessing partial observability in network sensor location problems.” *Transportation Research Part B: Methodological* 70: 65 – 89.
- Wales, David J., and Jonathan P. K. Doye. 1997. “Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms.” *The Journal of Physical Chemistry A* 101 (28): 5111–5116.

## Appendix A. Derivation of the Gradient of the Objective Function

In the following we show how the gradient of the OF  $F$  is derived. To do this we only show how the gradient of the first term is computed, as the derivation of the gradient of the second term is very similar. The gradient of the OF  $F$  is defined as:

$$\frac{\partial F}{\partial x_{ws}} = \omega_1 \cdot \frac{\partial f^{(1)}}{\partial x_{ws}} + \omega_2 \cdot \frac{\partial f^{(2)}}{\partial x_{ws}}, \quad \forall w \in W, s \in S. \quad (\text{A1})$$

To proceed we compute the gradient of the first term  $f^{(1)}$  as follows:

$$\frac{\partial f^{(1)}}{\partial x_{ws}} = \frac{\partial}{\partial x_{ws}} \left( \frac{\left( \sum_{\substack{w' \in W \\ s' \in S}} (x_{w's'} - \tilde{x}_{w's'})^2 \right)^{1/2}}{\left( \sum_{\substack{w' \in W \\ s' \in S}} \tilde{x}_{w's'}^2 \right)^{1/2}} \right) \quad (\text{A2})$$

We notice that the expression in the denominator is simply a constant. We can factor this constant out and get:

$$= \frac{\partial}{\partial x_{ws}} \left( \left( \sum_{\substack{w' \in W \\ s' \in S}} (x_{w's'} - \tilde{x}_{w's'})^2 \right)^{1/2} \right) \cdot \frac{1}{\left( \sum_{\substack{w' \in W \\ s' \in S}} \tilde{x}_{w's'}^2 \right)^{1/2}} \quad (\text{A3})$$

We then proceed by applying the *chain rule* to this expression and get:

$$= \frac{1}{2} \cdot \underbrace{\left( \sum_{\substack{w' \in W \\ s' \in S}} (x_{w's'} - \tilde{x}_{w's'})^2 \right)^{-1/2}}_{\text{Deriv. of outer func.}} \cdot \underbrace{2 \cdot (x_{ws} - \tilde{x}_{ws})}_{\text{Deriv. of inner func.}} \cdot \frac{1}{\left( \sum_{\substack{w' \in W \\ s' \in S}} \tilde{x}_{w's'}^2 \right)^{1/2}} \quad (\text{A4})$$

When taking the partial derivative of the inner function w.r.t.  $x_{ws}$  we notice that the terms with  $w' \neq w$  and  $s' \neq s$  are 0. We are thus only left with a single term, when taking the derivative of the inner function. Finally, we can reduce this expression and define the gradient of  $f^{(1)}$  as:

$$g_{ws}^{(1)} = \frac{\partial f^{(1)}}{\partial x_{ws}} = \frac{x_{ws} - \tilde{x}_{ws}}{\left( \sum_{\substack{w' \in W \\ s' \in S}} (x_{w's'} - \tilde{x}_{w's'})^2 \right)^{1/2} \cdot \left( \sum_{\substack{w' \in W \\ s' \in S}} \tilde{x}_{w's'}^2 \right)^{1/2}} \quad (\text{A5})$$

The gradient of the second term  $f^{(2)}$ , which describes the marginal effect of a change in the demand variables on the arc counts, can be computed in a similar way. The only difference is that the relation in Eq. (9) should be used in the derivation as well.

## Appendix B. Machine Learning Algorithms

**Feed-Forward Networks (FFN):** A feed-forward network is a nonparametric model and a type of artificial neural network with a directed acyclic graph structure consisting of *units* arranged in layers: First an input layer, then possibly several *hidden layers*, and lastly, an *output layer*. Each of the units in a layer has links to the units in a subsequent layer, and each of the links has an associated weight that determines the strength and sign of the connection. In each unit, an *activation function* is applied to

an offset term (usually referred to as a bias term in the FNN terminology) plus the weighted sum of activations coming from the units in a previous layer. Depending on the layer, different activation functions are used. No activation function is used in the input layer and a *ReLU* ( $\phi(\cdot) = \max(0, \cdot)$ ) activation function is usually used in the units situated in the hidden layers. The choice of activation function used in the output layer is tied to the nature of the task at hand. For example, a linear activation function is used for the regression task at hand where the target variables are continuous. A loss function that measures a model’s goodness-of-fit also needs to be defined to determine the bias terms and the weights of the links between the units. Just as the activation function used in the output layer is tied to the task at hand, the choice of the loss function is also closely tied to the activation function used in the output layer. For example, a Mean Squared Error (MSE) loss function is used for a regression task.

A FNN architecture is defined by the number of layers (the depth of the network  $m_d \in \mathbb{N}$  not counting the input layer) and the number of units in each layer (the width of a layer  $l_i \in \mathbb{N}$ , with  $i \in \{1, \dots, m_d\}$ ), along with the activation functions used in the different layers. In this case, to train the network, the problem is to find a set of weight matrices  $\mathcal{W} = \{\mathbf{W}_1, \dots, \mathbf{W}_n\}$  and corresponding biases  $\mathcal{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$  that minimize the Mean Squared Error (MSE) loss function:

$$(\hat{\mathcal{W}}, \hat{\mathcal{B}}) \in \arg \min_{\mathcal{W}, \mathcal{B}} \|\mathbf{y} - \hat{\mathbf{y}}_{\mathcal{W}, \mathcal{B}}(\mathbf{x})\|_2^2 + \lambda \sum_{i=1}^n \|\mathbf{W}_i\|_2^2 \quad (\text{B1})$$

$$\hat{\mathbf{y}}_{\mathcal{W}, \mathcal{B}}(\mathbf{x}) = (h_{m_d} \circ h_{m_d-1} \dots \circ h_1)(\mathbf{x}) \quad (\text{B2})$$

where  $h_i(\mathbf{a}) = \phi_i(\mathbf{W}_i^\top \mathbf{a} + \mathbf{b}_i)$  for  $i \in \{1, \dots, m_d\}$ . Furthermore, the activation function  $\phi_i$  is applied *element-wise* to the vector-valued argument, and  $\mathbf{a}$  contains all the activations received in the  $i$ th layer, while  $\mathbf{W}_i$  contains the corresponding weights, and  $b_i$  is the bias term. Finally, in the last term of Eq. (B1)  $\lambda \in \mathbb{R}_+$  is a regularization parameter added to reduce overfitting and improve the out-of-sample predictive performance. An appropriate value for the regularization parameter is usually chosen using out-of-sample cross-validation techniques.

Gradient-based methods are widely used for solving the problem in Eq. (B1). This is because gradients can be computed efficiently using the *backpropagation* algorithm (see, e.g., Goodfellow, Bengio, and Courville (2016) for an in-depth explanation). Essentially, we learn an input-output relationship by applying a sequence of semi-affine nonlinear transformations, which in terms of the layers in a FNN can be understood as applying the composite map in Eq. (B2) to a given input  $\mathbf{x}$ .

**$k$ -Nearest Neighbors ( $k$ -NN):**  $k$ -NN is one of the simplest nonparametric models used for regression. To predict the value  $\hat{\mathbf{y}}$  of a new unseen point  $\mathbf{x}$ ,  $k$  needs to be specified. Its value specifies the number of points from a training set  $\mathcal{D}$  closest to  $\mathbf{x}$  that should be used to calculate the predicted value  $\hat{\mathbf{y}}$ . Let  $N_k(\mathbf{x}) \subseteq \mathcal{D}$  be the set of  $k$  points closest to  $\mathbf{x}$  in terms of the Euclidean distance, then the predicted value of  $\mathbf{x}$  is calculated as the average of the nearest points:

$$\hat{y}_k(\mathbf{x}) = \frac{1}{k} \cdot \sum_{(\mathbf{x}', \mathbf{y}') \in N_k} \mathbf{y}'. \quad (\text{B3})$$

Unlike the FNN and most other ML algorithms, no model must be determined, rather the training data are kept and continuously used.