Dynamic Origin-Destination Matrix Estimation in Urban Traffic Networks

Nicklas Sindlev Andersen, Marco Chiarandini, Kristian Debrabant

Department of Mathematics and Computer Science
University of Southern Denmark
Odense, Denmark
{sindlev|marco|debrabant}@imada.sdu.dk

February 2, 2022

Abstract

Given the counters of vehicles that traverse the roads of a traffic network, we aim at reconstructing 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. In the inner level, given a tentative travel demand, we solve a dynamic traffic assignment problem to decide the routing of the users between their origins and destinations. In the outer level, we adjust the number of trips and their origins and destinations, aiming at minimizing the discrepancy between the consequent counters generated in the inner level and the given vehicle counts measured by sensors in the traffic network. We solve the dynamic traffic assignment problem 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 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 study 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 study machine learning algorithms to learn a model of the simulator that can then be used as a surrogated objective function in the optimization problem. We compare these approaches computationally on an artificial network. The gradient-based approaches perform the best in terms of archived solution quality and computational requirements, while the results obtained by the machine learning approach are currently less satisfactory but provide an interesting avenue of future research.

Keywords— Bi-level optimization, Dynamic origin-destination matrix estimation, Dynamic traffic assignment, Simulation-based optimization

Traffic on the road network of urban areas is continuously increasing. To keep high mobility standards, municipalities can act on the infrastructure, modifying it to avoid congestion and ensure reliability. Therefore, there is an urge to provide decision-makers with tools to make well-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 should be known to some degree of 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 travel 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 the road infrastructure and each other. However, congestion may occur, and users might change their decisions and re-distribute on the network. If the travel 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 travel demand is actually unknown, it is desirable to estimate the travel demand that exists in a particular traffic network.

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 trying to estimate dynamic OD matrices is usually referred to as the Dynamic OD Estimation problem (DODE) in the literature. Different formulations exist but it is commonly and traditionally 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 in the inner problem and a Dynamic Traffic Assignment (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 aiming at minimizing 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 in every road of the network as measured by sensors or trackers. 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. 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, for example, by the number of people living in the zones that are considered as origins and destinations.

Solving bi-level optimization problems is, in general, 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.

Inspired by recent research, we investigate and solve the DODE problem using Machine Learning (ML) approaches that leverage problem-specific knowledge. More precisely, 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 optimization problem. To benchmark the machine learning approaches, we compare them combined with classical gradient-based approaches. We provide an analysis of the advantages and disadvantages of these approaches. Overall, our work provides the following contributions:

- We investigate a possible way to use machine learning in the solution of the DODE problem by learning a model for the inner problem and bypass its computational cost.
- We perform a computational assessment of our proposal compared with two classical gradient-based approaches under a controlled environment made possible by a careful design of a synthetic scenario.
- Although the performance of the machine learning approach will turn out disappointing, we present a fair comparison of classical gradient-based approaches re-implemented and compared on the same controlled ground that provides interesting results on its own.

We use the well-established, open-source traffic simulator SUMO [1] 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 2.

The rest of the paper is outlined as follows. In Section 1, we give a general overview of the existing literature on the DODE problem. In Section 2, we introduce notation and the traditional bi-level optimization formulation of the DODE problem that we will use in the subsequent sections. Then, in Section 3, we describe the studied approaches that are applied for the solution of the problem. More precisely, the studied approaches are applied to a case study on a small irregular grid network where, in this context, an in-depth analysis is provided. Finally, in Section 6, we conclude the work by commenting on the results.

1 Literature Review

In the classical static OD estimation problem, static OD trips are considered and the problem is to estimate the travel demand in a single time interval (a single OD matrix is estimated). On the other hand, in the DODE problem, time-dependent OD trips are considered, and the problem is to estimate the travel demand in several time intervals (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. Moreover, many of the ideas and insights that have been gained through studying the static OD
estimation problem are also extended and applied to the dynamic problem. For an overview of the different classical solution approaches to this problem see [2, 3]. On the other hand, the most recent and extensive review of existing literature on the DODE problem is provided in [4, 5], while a literature review on much earlier work is provided in [6]. 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 [7]. Finally, an overview of many related and relevant problems is provided in [8].

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. With this distinction in mind, we focus on the literature concerned with DTA-based solution approaches.

Usually, solution approaches to the bi-level formulation of the DODE problem adopt and investigate general-purpose optimization algorithms. 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 solutions 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 only rely solely on objective function evaluations. Here, the objective function is the discrepancy between the quantities determined by a DTA solution and the corresponding observed quantities given as input.

Within the class of gradient-based algorithms we can further categorize assignment matrix-based or assignment matrix-free approaches. This distinction can be made based on whether an assignment matrix is used as a part of the applied optimization algorithm or not. An assignment matrix summarizes the result of a DTA in that it 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 thus means that the assignment matrix varies as a function of the travel demands. The solution approaches to the DODE problem that uses a gradient and assignment matrix-based approach exploit the fact that it can be possible to analytically derive the exact gradient in some instances, where a functional relationship between the variables that are to be estimated can be established through the use of an assignment matrix. On the other hand, a gradient-based and assignment matrix-free approach primarily uses finite difference approximations of the gradient, which are established through objective function evaluations.

The most recent studies that adopt a gradient and assignment matrix-based approach are given in [9, 10, 11, 12, 13]. 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 assignment matrix-based approaches is that the only type of traffic observations usually given as input to the optimization problem are count observations. The modeling of the relationship between the variables to be estimated is done through the assignment matrix; in this case, there is a limit to the type of observations that can be accommodated in the optimization problem using this type of approach. For example, the functional relationship between travel demands and count observations can easily be established if a proportional relationship between these variables is assumed (see Eqn.10). For other types of observations, it can be harder and more cumbersome to determine appropriate relationships between the variables to be estimated.

Gradient and assignment matrix-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 [14] tailoring it for the DODE problem. This algorithm only relies on objective function evaluations with no need for an explicit characterization of the relationship between the variables to be estimated. In each iteration of the SPSA algorithm, objective function evaluations are used to approximate the gradient. This approach makes it possible to include other types of traffic observations beyond just count observations and it has therefore been the predominant approach in these cases [15, 16, 17, 18, 19, 20]. 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 objective function shape and gradient. Different enhancements to the SPSA algorithm in the listed 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.

If a model for the objective function is not available by means of theoretical argumentation, a surrogate
model can be estimated on the basis of empirical data. The surrogate model can then be optimized by
gradient based or derivative-free methods. In [6], response surface techniques are used to model the objective
function by low order polynomial functions fitted locally to the values in correspondence of sample points of the
search space. The author applies a Stable Noisy Optimisation 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 and extends the well-known Nelder-Mead algorithm. However, in the context of model-based
approaches gradient-based methods seem to perform better than derivative-free methods [21]. In [6] the hybrid
Box-Complex and SNOBFIT algorithm was found to perform worse both in efficacy and scalability with respect
to the SPSA algorithm while in [15] an evolutionary algorithm was also shown inferior to the SPSA algorithm.

Generally, the derivative-free solution algorithms require a high number of objective function evaluations
to obtain results that are comparable to the gradient-based solution algorithms. In this context, a promising
line of research is provided in [22], which describes a meta-modeling-based approach, that promises good
computational results under tight computational budgets. The meta-modeling approach is compared against
the SPSA and a derivative-free pattern search algorithm. The meta modeling-based approach performs well,
while the pattern search algorithm and the SPSA algorithm perform poorly in comparison.

Our work presented in Section 3.2 contributes to this thread of research by studying machine learning
approaches to determine surrogate models as an alternative to response surface techniques [6].

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 objective function evaluations and exhibit the advantage that
they can be distributed and parallelized, as they allow independent objective function evaluations. Evolutionary
algorithms have been the most popular among derivative-free optimization algorithms. Notably, in [23] an
evolutionary algorithm was applied in a setting with distributed and parallel computing. Other studies that
apply evolutionary algorithms can be found in [15, 24], while [25] also use an evolutionary algorithm, but in a
non-DTA based approach.

Finally, we note that two variants of DODE problems are addressed in the literature. They are characterized
by whether their offline or online setting. In an offline setting, historical data is used to estimate the network-
wide travel demand, such that long-term predictions of future traffic conditions can be made. As mentioned
earlier, this is especially beneficial to decision-makers, as it enables them to evaluate different planning scenarios
and make informed decisions when proposed changes to infrastructure and facilities are made. Offline travel
demand estimation procedures are usually applied in network-wide studies, which means that the allowed
computational cost can be considerably high in this setting. In an online setting, real-time traffic data is used,
in addition to historical data, to estimate the current level of travel 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 travel 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
applied. For references on work that study the online DODE problem, we refer the reader to [26, 27, 4]. In
this work, we focus on the DTA-based, offline, bi-level formulation of the DODE problem.

2 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. It is thus possible to have an arc between two nodes, where no opposite-going arcs exist.

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 pair $w = (i, j) \in W = O \times D$. Here $W$ is the set of all origin-destination 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 $Q = \{1, \ldots, n_Q\} \subseteq A$.

To define the main quantities of interest in the DODE problem, we introduce a time period of analysis $T = [0, t_{end}]$ discretized into a set of $n_T$ subintervals $\{(0, t_1), (t_1, t_2), \ldots, (t_{n_T-1}, t_{n_T} = t_{end})\}$ of equal duration. For the sake of convenience, we identify these intervals by the indices in the set $S = \{1, \ldots, n_T\}$. These indices define the time intervals in which we want to estimate the travel demands for each of the $m$ OD pairs. The
estimation of the travel 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 $A \in \mathbb{R}^{p_1 \times p_2}$, we concatenate consecutive rows of the matrix in a column vector $(\mathbb{R}^{p_1 \times p_2} \rightarrow \mathbb{R}^{p_1 \times p_2})$:

$$\text{vec}(A) = a = [a_{11}, \ldots, a_{p_11}, \ldots, a_{12}, \ldots, a_{p_12}, \ldots, a_{p_1p_2}]^T \in \mathbb{R}^{p_1 \times p_2}.$$  (1)

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

- $x, \hat{x} \in \mathbb{R}^{m \times S}$ are the vectors of estimated demands and seed demands, respectively. Each element of these vectors $x_{us}$ or $\hat{x}_{us}$, respectively, is associated with an OD pair $w \in W$ and a time interval $s \in S$.

The vector $x$ of estimated demands contains the estimate of the number of trips made for each OD pair in each of the time intervals. The vector $\hat{x}$ of seed demands defines the prior knowledge of the number of trips made for each OD pair in each of the time intervals. This prior knowledge is usually assumed to have been obtained from a previous travel demand study, e.g., from a study where a population survey was undertaken. Note that we model discrete values as real numbers we thus define the continuous relaxation of a discrete optimization problem.

- $x^{\text{Lower}}, x^{\text{Upper}} \in \mathbb{R}^{m \times S}$ are the vectors of lower and upper bounds on the estimated demands, respectively. These upper and lower bounds define the search space.

- $c, \hat{c} \in \mathbb{Z}^{n \times Q \times S}$ are the vectors of estimated and observed arc counts, respectively. Each element of these vectors $c_{qs}$ or $\hat{c}_{qs}$, respectively, is associated with a sensor $q \in Q$ and a time interval $s \in S$.

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

$$\min \quad F(x, \hat{x}, c(x), \hat{c}) \quad \text{(2)}$$
subject to

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

The solution of this problem, $x^*$, is the travel demand estimate that results from minimising the objective function $F$, which we define as the weighted sum of the measures of discrepancy between the estimated quantities and their corresponding observed or a priori values:

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

In our specific case, the discrepancy $f^{(1)}$ is measured between the estimated travel demands $x$ and a priori known travel demands $\hat{x}$, and the discrepancy $f^{(2)}$ between the estimated arc counts $c(x)$ and the observed arc counts $\hat{c}$. Note that in a more general setting, the objective function $F$ may consist of several additional function terms $f^{(3)}, f^{(4)}, \ldots$ with respective weights $\omega_3, \omega_4, \ldots$ that take into account additional available information, such as arc speed, arc density, travel time, turning ratios, etc. By including additional information into the estimation problem, one can hope to improve the guidance of the search process.

To evaluate the objective function in Eqn. (2) for a proposed vector $x$ of travel demands, an inner optimization problem must be solved to find a corresponding vector $c(x)$ of arc counts. The inner optimization problem takes the form of a DTA problem, where the task is to determine the routing of users between origins and destinations according to a certain assignment principle. The users of the traffic network are assumed to make decisions in accordance with the criterion specified by the assignment principle. Two examples of basic 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 [28]), which states that each user makes decisions that minimize his/her own individual travel time, and (ii) the System Optimum (SO) principle (Wardrop’s second principle [28]), which states that users make joint decisions to minimize the total system travel time.

Dynamic traffic assignment models adopt an assignment principle and incorporate several different travel choice components to reflect the travel choices a user of a real 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 [29], 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. The mesoscopic model is computationally cheaper to solve and is able to provide the necessary data to a sufficient degree of detail needed in the estimation problem. Therefore, our choice fell on 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 respect an assignment principle. In SUMO, this assignment principle is the Stochastic User Equilibrium (SUE). The SUE assignment principle extends the UE assignment principle, where stochastic elements have been incorporated in 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 [30]. This method determines the probabilities of choosing between certain 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.

In this context, we note that a single traffic simulation is single-threaded and sequential, while the routing of the users in the network, i.e., the selection of the least-cost paths (in terms of the path travel time) between origins and destinations can be made in parallel. In SUMO, by default, the least-cost paths are computed by a time-dependent version of Dijkstra’s algorithm.

Note that we modeled travel demands \( x \) as continuous variables while the mesoscopic simulator implemented by SUMO solves a discrete optimization problem. In this context, when the travel demands are given to SUMO as input, they are rounded to the nearest integer values. In all other cases, the travel 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:

\[
\Gamma : \mathbb{R}_+^{m \times n} \rightarrow \mathbb{Z}_+^{n \times m}.
\]  

This function implicitly indicates that a DTA needs to be performed given input travel demands \( x \). As a result of the DTA, we obtain arc counts \( c(x) = \Gamma(x) \) as output. The arc counts \( c(x) \), together with the corresponding input travel demands \( x \), can then be used in the objective function in Eqn. (4) to determine the quality of the current travel demand estimate.

Eqn. (3) defines upper and lower bounds on the travel demands. These are usually supplied to avoid a situation where an unrealistic high amount of travel demand is loaded on the traffic network, exceeding the network’s capacity. In a realistic setting, the bounds on the travel 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 \( f(x) \) for the objective function defined in Eqn. (4), as \( \hat{x} \) and \( \hat{c} \) are given and \( c(x) \) is derived from \( x \).

### 3 Solution Approaches

Common to all our solution approaches are the exact functional form of the functions \( f^{(1)} \) and \( f^{(2)} \) used in the objective function \( F \) in Eqn. (4), and the constraints that directly limit the search space of the main variables to be estimated, i.e., \( x \).

Several different definitions of the discrepancy between observed and estimated quantities in the objective function \( F \) are possible. Drawing upon the knowledge obtained in [16] and [8], we use an objective function that penalizes large errors between observed and estimated quantities while weighting the different objective function terms evenly, i.e., we take into account that the different quantities \( f^{(1)} \) and \( f^{(2)} \) that enter in the objective function can have different magnitudes and hence normalize their values. Thus, we define the terms \( f^{(1)} \) and \( f^{(2)} \) used in the objective function \( F \) in the following functional form:

\[
f^{(1)}(x, \hat{x}) = \sqrt{\frac{\sum_{w \in W} \sum_{s \in S} (x_{ws} - \hat{x}_{ws})^2}{\sum_{w \in W} \sum_{s \in S} \hat{x}_{ws}^2}}
\]  

(6)
and

$$f^{(2)}(c(x), \hat{c}) = \frac{\sqrt{\sum_{q \in Q} \sum_{s \in S} (c_{qs} - \hat{c}_{qs})^2}}{\sqrt{\sum_{q \in Q} \sum_{s \in S} \hat{c}_{qs}^2}}$$

To further limit the search space (beside the upper and lower bounds on the travel demands), we introduce generation constraints (also referred to as production constraints) [8], that set an upper bound on the number of outbound trips from an origin. More precisely, we let $o_i$ denote the maximum number of vehicles that can leave origin $i \in O$ and enforce that:

$$\sum_{j \in D, w=(i,j) \in W} x_{ws} \leq o_i \quad \forall i \in O.$$  \hfill (8)

This type of constraints can easily be incorporated in the problem formulation (as mentioned in [16]) and it is typically the case that data for the values of $o_i$, $i \in O$ is readily available and highly reliable, e.g., from survey data or population density statistics. Other possible constraints are also mentioned in [8], 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. Data pertaining to these types of 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 $\tau \in \{0, \ldots, \tau_{\text{max}}\}$ where $\tau_{\text{max}}$ is the maximum number of iterations of an algorithm. We thus let $x_{\tau}$ define the estimate at iteration $\tau$.

### 3.1 Gradient-based Approaches

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

$$x_{\tau + 1} = x_{\tau} + \eta_{\tau} \cdot d_{\tau}.$$  \hfill (9)

Here $\eta_{\tau}$ is a scalar-value that we will refer to as the step length and $d_{\tau}$ is the descent direction. Different gradient-based optimization algorithms differ in how the descent direction and the step length are chosen. For the DODE problem, as mentioned in Section 1, it is possible to distinguish between assignment matrix-based and assignment matrix-free gradient-based approaches.

#### 3.1.1 An Assignment Matrix-based Approach

Through the use of an assignment matrix the gradient of the objective function can be calculated analytically. The (vectorized) assignment matrix $p_{\tau} = \text{vec}(P_{\tau}(x_{\tau}))$ at an iteration $\tau$ consists of entries $p_{qrws} \in [0, 1]$ 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:

$$c_{qr} = \sum_{w \in W, s \in S, s \leq r} p_{qrws} \cdot x_{ws}, \quad \forall q \in Q, r \in S.$$  \hfill (10)

Through the use of the assignment matrix it is possible to derive an analytical expression for the gradient of the objective function $F$ at every iteration:

$$\frac{\partial F}{\partial x} = \omega_1 \cdot g^{(1)} + \omega_2 \cdot g^{(2)}.$$  \hfill (11)
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:

$$
\begin{align*}
\mathbf{g}_w^{(1)} &= \frac{\partial f^{(1)}}{\partial x_w} \\
&= \frac{x_w - \bar{x}_w}{\sqrt{\sum_{w' \in W} (x_{w's} - \bar{x}_{w's})^2}}, \quad \forall w \in W, s \in S,
\end{align*}
$$

(12)

$$
\begin{align*}
\mathbf{g}_w^{(2)} &= \frac{\partial f^{(2)}}{\partial x_w} \\
&= \frac{\sum_{q' \in Q, r' \geq s} (\hat{c}_{q'r'} - \hat{c}'_{q'r'}) \cdot p_{q'r'w}}{\sqrt{\sum_{q' \in Q, r' \geq s} \hat{c}'_{q'r'}^2}}, \quad \forall w \in W, s \in S.
\end{align*}
$$

(13)

The descent direction in Eqn. (9) is then simply the negative of the expression in Eqn. (11):

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

(14)

### 3.1.2 An Assignment Matrix-free Approach

As mentioned in Section 1, most of the studied gradient-based and assignment matrix-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 assignment matrix is needed in the derivation of the gradient of the first objective function term $f^{(1)}$, we can simply use the exact gradient defined in Eqn. (12). However, the gradient of the second objective function term $f^{(2)}$ will have to be approximated because now no assignment matrix is given and no functional relationship between the variables to be estimated is assumed.

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

$$
\mathbf{g}_r^{(2)} = \frac{f^{(2)}(\mathbf{c}(\mathbf{x}), \hat{\mathbf{c}}) - f^{(2)}(\mathbf{c}^-(\mathbf{x}), \hat{\mathbf{c}})}{b_{\tau} \Delta}, \quad \tau \in \{0, \ldots, \tau_{\max}\}.
$$

(15)

Here $\mathbf{c}(\mathbf{x}) = \Gamma(\mathbf{x}_r), \mathbf{c}^-(\mathbf{x}) = \Gamma(\mathbf{x}_r - b_{\tau} \cdot \Delta_{\tau})$ and $\Delta_{\tau} \in \{-1, 1\}^{m \cdot n_w}$ is a random perturbation vector, where the entries $\Delta_{w,s}, w \in W, s \in S$ are independent Bernoulli random variables taking the value $\pm 1$ with probability $1/2$. This distribution is also known as the Rademacher distribution. Furthermore, the vector $\Delta_{\tau}$ consists of entries $1/\Delta_{w,s}$ and $b_{\tau} = b/\tau^{-\gamma}$ is defined with respect to the parameters $b > 0$ and $\gamma > 0$. These parameters and the Rademacher distribution are usually chosen because they satisfy certain conditions [31] that ensure the algorithm to be able to converge to a minimum, asymptotically. As we are not interested in the asymptotic behavior of the algorithm but want to obtain travel demand estimates in a small number of iterations while ensuring compatibility with SUMO, we keep the perturbation parameter fixed $b_{\tau} = 1$ throughout all iterations. This is equivalent to perturbing each element in the demand vector by $\pm 1$ trip.

### 3.1.3 Determining Step Lengths

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

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

(16)

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

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

(17)

Then, we determine:
1. An upper bound \( \ell \) on the step length \( \alpha \) imposed by the generation constraints in Eqn. (8). An algorithmic sketch to carry out this task is given in Algorithm 1.

2. A number of equally spaced points \( \alpha_1, \ldots, \alpha_n, n \geq 2 \) to evaluate in the interval \([\ell/n, \ell]\).

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

\[
(\alpha_0, y(\alpha_0)), \ldots, (\alpha_n, y(\alpha_n)).
\]

The point \((\alpha_0 = 0, y(\alpha_0) = F(x_\tau))\) is already known. In this case, by specifying two additional sample points we are able to uniquely fit a second degree polynomial. More precisely, an estimator for the function \(y(\alpha)\) in Eqn. (17) is:

\[
\hat{y}(\alpha) = \beta_0 + \beta_1 \cdot \alpha + \beta_2 \cdot \alpha^2
\]

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

**Algorithm 1:** Procedure for determining the upper bound to the step length in the descent direction.

```plaintext
1 Procedure DetermineMaxStepLength(x_\tau, x_l, x_u, d_\tau, \ell, \Delta = 10^8, \epsilon_1 = 10^{-8})
2 \ell \leftarrow 0
3 while true do
4 \quad \ell \leftarrow \ell + \ell \Delta
5 \quad \hat{x} \leftarrow x_\tau + \ell \cdot d_\tau
6 \quad // Apply upper and lower bounds
7 \quad \hat{x} \leftarrow \max(x_l, \hat{x})
8 \quad \hat{x} \leftarrow \min(x_u, \hat{x})
9 \quad // Check if generation constraints are violated or whether
10 \quad // all upper or lower bound constraints are binding
11 \quad if \sum_{j \in D, w=(i,j) \in W_S} x_{ws} > o_i \forall i \in O \lor (x_{ws} \equiv x_{ws}^{\text{Lower}} \lor x_{ws} \equiv x_{ws}^{\text{Upper}}) \forall w \in W, s \in S
12 \quad then
13 \quad \quad // Backtrack and try again with a smaller step size increment \eta_\Delta
14 \quad \quad \ell \leftarrow \ell - \ell \Delta
15 \quad \quad \ell_\Delta \leftarrow \ell_\Delta / 2
16 \quad \quad if \ell_\Delta < \epsilon_1 then
17 \quad \quad \quad break
18 \quad \quad // Return the largest positive step length that does not violate any constraints
19 \quad \quad return \ell
```

### 3.1.4 Computational Complexity of the Gradient-based Approaches

If we look at the computational cost associated with the assignment matrix-based approach, we notice that two objective function evaluations and thus DTA problems are solved in each iteration of the approach. First, a single DTA is needed to obtain a (vectorized) assignment matrix \(p_\tau\) such that the descent direction and three of the best step length in the descent direction can be determined (described in Section 3.1.3). Then, one additional DTA is solved to evaluate the new point that has been found by using the best step length that minimizes the objective function in the descent direction.

On the other hand, the assignment matrix-free approach needs to perform up to four DTA problem solutions in each iteration, i.e., one DTA is required to estimate the gradient and three additional DTAs are required for determining the best step length in the descent direction. As with the assignment matrix-based approach an additional DTA is then needed to evaluate a new estimate that has been found.

### 3.2 Machine Learning Approaches

For a certain geographical area of study and traffic network, the DODE problem is solved frequently, whenever a more up-to-date travel 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 form of recent traffic observations is the only component of the DODE problem that changes from one optimization task to another. Other times, like in network design analysis the topology of the network may change while all the other elements remain the same. The gradient-based approaches that apply sequential optimization algorithms to solve the DODE problem imply performing many DTAs in each of the separate optimization tasks, essentially starting from scratch every time a new travel 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 valuable information is usually discarded and not used in subsequent optimization tasks. This is important information that could be used but was not, primarily due to the inherent sequential nature of the gradient-based approaches. In this context, a machine learning approach becomes very appealing as it allows leveraging data produced by many DTAs. This is done by learning some part of the optimization problem, making it possible to bypass the most demanding and time-consuming solution of the DTA problem. In other words, the machine learning approach enables information produced by previous DTAs and previous optimization tasks to be used in subsequent optimization tasks. In the following we model the machine learning task.

The DTA performed by the traffic simulator SUMO can be regarded as a black-box function that maps a certain specification of input travel demands to network-specific quantities, such as arc counts, speeds, densities, etc. Ultimately, these quantities can be produced by the simulator and describe the utilization and state of the traffic network resulting from a given input specification of travel demands. As mentioned in Section 2, we will primarily focus on the arc counts as real-world data pertaining to this type of traffic stream characteristic is widely used and readily available. In this case, the black-box function representing the DTA performed by SUMO can be described by the map in Eqn. (5), which is a vector-valued multivariate function that describes the input-output relationship of the DTA performed by SUMO with respect to the output quantity we are interested in modeling. The main idea behind the machine learning approach is essentially to learn a model of this input-output relationship.

To be able to learn a model of the input-output relationship an initial data set is needed. How this initial data set is obtained is facilitated through the use of a sampling strategy which is a technique for deciding how many and which points should be included in an initial data set provided to a learning algorithm. Through the use of the initial data set, the learning algorithm is trained in a supervised fashion which results in a model \( \hat{f} \) for the input-output relationship in Eqn. (5). Subsequently, the DODE problem can be solved by using the obtained model as a surrogate for DTA in the objective function that can be optimized as done earlier.

Overall, the machine learning approach put forth here consists of three primary components: a sampling strategy, the application of a machine learning 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.

3.2.1 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 work in terms of computational resources and time. In other words, for the initial data set 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 design, central composite design, different factorial designs, and
Feed-forward Networks: learning algorithms that are inherently multi-target is that they can take advantage of inter-target correlations.

In contrast to parametric models, nonparametric models are more flexible as no prior assumptions are made on the structure of the model. In our experiments (see Section 5) we will use an initial data set with points sampled according to the numbers in a Sobol sequence (an example of this sampling strategy, in the plane, is shown in Figure 1).

Formally, let \( D_0 = \{ (x_1, y_1), \ldots, (x_n, y_n) \} \) denote the initial data set where \( x_i \in \mathbb{R}^{n \times S} \) is an element of a Sobol sequence and \( y_i = \Gamma(x_i) \in \mathbb{R}^{Q \times S} \) for all \( i \in \{1, \ldots, n\} \). Recall that \( \Gamma \) represents the function applied by the SUMO simulator. The learning task is determining \( \hat{\Gamma} \) such that it approximates well \( \Gamma \) on new data points.

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

In our experiments (see Section 5) we will use an initial data set with points sampled according to the numbers in a Sobol sequence (an example of this sampling strategy, in the plane, is shown in Figure 1).

3.2.2 Machine Learning Algorithms

A wide selection of off-the-shelf implementations of general-purpose ML algorithms can be used to find \( \hat{\Gamma} \). We refer the interested readers to the book *The Elements of Statistical Learning* [33]. The machine learning 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.

If we are interested in learning an input-output relationship of a vector-valued multivariate function as in Eqn. (5), then several different ML algorithms that are inherently multi-target can be used. A multi-target ML algorithm can learn a single model that simultaneously predicts \( p \) target variables. An advantage of learning algorithms that are inherently multi-target is that they can take advantage of inter-target correlations. Inherently multi-target ML algorithms that we will make use of are described in the following:

**Feed-forward Networks:** 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 feed-forward network 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 \((\phi(\cdot) = \text{max}(0, \cdot))\) 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 feed-forward network 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 \( i \in \{1, \ldots, m_d\} \), 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 \( W = \{W_1, \ldots, W_n\} \) and corresponding biases \( B = \{b_1, \ldots, b_n\} \) that minimize the Mean Squared Error (MSE) loss function:

\[
(\hat{W}, \hat{B}) \in \arg \min_{W, B} ||y - y_{W, B}(x)||_2^2 + \lambda \sum_{i=1}^n ||W_i||_2^2
\]

\[
y_{W, B}(x) = (h_{m_d} \circ h_{m_d-1} \circ \ldots \circ h_1)(x)
\]

where \( h_i(\mathbf{a}) = \phi_i(W_i^T \mathbf{a} + b_i) \) for \( i \in \{1, \ldots, m_d\} \). 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 \( W_i \) contains the corresponding weights, and \( b_i \) is the bias term. Finally, in the last term of Eqn. (19) \( \lambda \in \mathbb{R}_+ \) 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 Eqn. (19). This is because gradients can be computed efficiently using the backpropagation algorithm (see, e.g., [34] 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 feed-forward network can be understood as applying the composite map in Eqn. (20) to a given input \( x \).

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

\[
\hat{y}_k(x) = \frac{1}{k} \sum_{(x', y') \in N_k} y'.
\]  

Unlike the feed-forward neural network and most other machine learning 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 feed-forward neural network or k-NN model and estimate its performance on unseen data, we perform hyperparameter tuning through an exhaustive grid search and use k-fold cross-validation (note here that this \( k \) is different from \( k \) in k-NN) on an initial data set. For a certain hyperparameter configuration, k-fold cross-validation consists of the following steps: (i) Divide the data set into \( k \) non-overlapping groups of approximately equal size, then (ii) save a single fold as a validation data set and use a machine learning 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. Step (i) and (ii) are repeated \( k \) times where a new fold is treated as the validation data set. Through this procedure, we obtain error estimates \( MSE_1, MSE_2, \ldots, MSE_k \) that can be used to calculate the overall k-fold cross-validation score:

\[
CV_k = \frac{1}{k} \sum_{i=1}^{k} MSE_i.
\]  

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 \( CV_k \) score is chosen in a model selection context. The value of \( k \) is commonly set to 5 or 10, usually due to computational considerations [35].

### 3.2.3 Optimization

We can then solve the upper-level optimization problem in Eqn. (2) with \( \Gamma \) to model the relationship in Eqn. (5). So instead of obtaining arc counts by performing a time-consuming DTA with SUMO, we obtain arc counts by querying the machine learning model \( \hat{\Gamma} \). Since now evaluating the objective function has become fast, 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 [36] and implemented by the SciPy Python library [37]. 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 objective function value.

The basinhopping algorithm extends the idea of the simulated annealing optimization algorithm by applying a local optimization algorithm to the perturbed point to then find a local minimum of the objective function 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 Python library [37]. The reason for this choice is that this algorithm can handle not only the upper and lower bound constraints in Eqn. (3) but also the inequality constraints in Eqn. (8).

Ultimately, the basinhopping algorithm is applied to find an approximate minimum of the DODE problem. The final travel demand estimate will then be evaluated by SUMO as a final result.
4 Experimental Setup

We decided to test and compare the alternative 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 time period of analysis $T$, and (iii) a ground-truth vector of travel demands on the basis of (i) and (ii). The specific parameters that were used in each of the experiments are described in Section 5. Here, we focus on details pertaining to components (i)-(iii).

The network can be defined as an irregular grid network. The irregular grid network was generated such that it contains 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 irregular grid network that was used is depicted in Figure 2a 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 a total of 132 OD pairs. Furthermore, the arcs in the network were prescribed a speed limit of 50 kilometer/hour and given a length that results from perturbing the coordinates of each of the nodes in the network by a random number.

We settled on a one-hour scenario with a time period of analysis $T = [0, t_{\text{end}})$, 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 $S = \{1, \ldots, n_S = 4\}$, where each of the indices in this set refers to a time interval of 900 seconds = 15 minutes duration. Given this information and a traffic network with OD pairs $W$, a ground-truth vector of travel 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 $b$:

$$x_{ws}^{\text{True}} \sim U(a, b), \forall w \in W, s \in S. \quad (23)$$

From the ground-truth vector of travel demands, trip productions $o_i$ for $i = 1, \ldots, |O|$ that enter into the generation constraints (in Eqn. (8)) were obtained in addition to upper and lower bounds on the estimated travel demands:

$$x_i = 0 \quad \text{and} \quad x_u = 1.5 \cdot \max(x) \cdot 1, \quad (24)$$

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

Finally, the ground-truth vector of travel demands was given to SUMO as input and the iterative DTA procedure implemented by SUMO was run for 15 iterations yielding the vector $\hat{x}$ of observed arc counts.

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

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

$$\tilde{x}_{ws} = x_{ws}^{\text{True}} \cdot (0.7 + 0.3 \cdot u_{ws}), \quad u_{ws} \sim U(0, 1), \forall w \in W, \forall s \in S. \quad (25)$$

**Prior high-demand vector (HD):** this scenario simulates the case were slightly higher travel demands are available from a previous study:

$$\tilde{x}_{ws} = x_{ws}^{\text{True}} \cdot (0.9 + 0.3 \cdot u_{ws}), \quad u_{ws} \sim U(0, 1), \forall w \in W, \forall s \in S. \quad (26)$$

These two scenarios were constructed based on the guidelines provided in [8] and the computational experiments performed in [16, 13].

The ground-truth vector of travel demands was generated according to Eqn. (23), using lower bound $a = 1$ and upper bound $b = 20$. An appropriate 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 decided experimentally by sampling different ground-truth vectors of travel 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 travel demands with corresponding upper bound $b$ that generated the most realistic test data was chosen as the basis for the case study using this network.

The amount of congestion present in the network was determined based on the amount of time vehicles are unable to drive at free-flow speed. The time vehicles lost due to congestion, i.e., time lost due to driving below free-flow speed, along with other characteristics of the synthetically generated test scenario, is presented in Figure 2b.

For the results we present in the following sections, it is assumed that sensors cover 100% of the network. In addition to this, the arc counts obtained by the sensors are assumed to be error-free. This is, of course, not
(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.

the case in a real traffic network, but it greatly simplifies the analysis and comparison of the different solution
approaches. 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 [38, 39, 40]. Lastly, we mention that
all the experiments were executed using an initial feasible solution consisting of all ones, i.e., \( x_{\tau = 0} = 1 \). This
initial solution was chosen so as not to confine the search for a minimum to the vicinity of the seed demands
and to remove a possibly confusing factor, the initial solution.

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

\[
RMSE(y, \hat{y}) = \left( \frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_j)^2 \right)^{\frac{1}{2}}.
\]  

(27)

This error measure is easily interpretable because the error is given in the same units as the data. Fur-
thermore, the RMSE is reported separately for each of the different quantities that enter into the optimization
problem, i.e., we report the RMSE statistic for the arc counts and the travel demands. Finally, we remark that
for the travel demands, the RMSE is calculated based on the ground-truth values and not on the seed travel
demand vectors that enter in the optimization problem.
Table 2: An overview and summary of the results obtained in the different experiments E1-E12. Here, we define M1: Assignment matrix-based approach, M2: Assignment matrix-free approach, M3: $k$-NN predicting $\Gamma$, M4: Feed-forward neural network predicting $\Gamma$.

| Method | Experiment | Seed Matrix | Total OF Evaluations | RMSE($x$, $x^\text{True}$) | RMSE($c(x)$, $\hat{c}$) |
|--------|------------|-------------|----------------------|-----------------------------|--------------------------|
| M1     | E1         | None        | 201                  | 1.3413                      | 9.9426                   |
|        | E2         | LD          | 201                  | 1.3413                      | 9.9426                   |
|        | E3         | HD          | 201                  | 1.2516                      | 9.0468                   |
| M2     | E4         | None        | 203                  | 9.0188                      | 19.4008                  |
|        | E5         | LD          | 201                  | 2.0346                      | 13.5121                  |
|        | E6         | HD          | 202                  | 6.7992                      | 16.1219                  |
| M3     | E7         | None        | 201                  | 11.1247                     | 27.0574                  |
|        | E8         | LD          | 201                  | 9.6493                      | 20.8409                  |
|        | E9         | HD          | 201                  | 9.6493                      | 20.8409                  |
| M4     | E10        | None        | 201                  | 9.6494                      | 20.8409                  |
|        | E11        | LD          | 201                  | 9.6494                      | 20.8409                  |
|        | E12        | HD          | 201                  | 9.6494                      | 20.8409                  |

Table 4: The machine learning model hyperparameters chosen for hyperparameter tuning. Preliminary experiments were performed to narrow down the domains of the hyperparameters.

| Machine Learning Model       | Hyperparameter Name | Domain |
|------------------------------|---------------------|--------|
| Feed-forward neural network  | Architecture       | \{0.75 \cdot m \cdot n_S, 0.50 \cdot m \cdot n_S, 0.50 \cdot m \cdot n_S, 0.25 \cdot m \cdot n_S, 0.25 \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 $k$      | \{2^i : z = 2 + i \cdot (6-2)/14, \forall i = 0, \ldots, 14\} |

5 Experimental Results

We present an overview of the experiments executed in together with summary results in Table 2.

To compare the different approaches fairly a soft upper bound on the number of objective function evaluations was set to 201, essentially meaning each of the approaches is allowed to use SUMO for about 201 DTAs. However, due to the inherent nature of some of the applied approaches, a few more objective function evaluations and thus DTAs might be performed in an iteration and the total number of objective function evaluations might not sum up to 201 exactly.

The gradient-based approaches, denoted by M1 for the assignment matrix-based approach and by M2 for the assignment matrix-free approach in Table 2, do not need parameter setting before tackling the DODE problem. This is the case, as the one parameter that would typically have, that is, the step length occurring in the generic Eqn. (9), is determined automatically by the solution of the line search problem in Eqn. (16). Thus, these algorithms are pretty straightforward to apply and no specific consideration has to be made concerning the parameters.

The final results obtained from the gradient-based approaches are plotted in Figure 3a and Figure 4a. In these figures, the estimated quantities $x$ and $c(x)$ have been plotted against the ground-truth vector of travel demands $x^\text{True}$ and the corresponding vector of observed arc counts $c(x)$, respectively. The corresponding plots that show the objective function value history of the two algorithms are shown in Figure 3b and Figure 4b.

If we compare the results in Figure 3a and Figure 4a between the two gradient-based approaches, we observe that the assignment matrix-based approach, in general, out-performs the matrix-free approach. In fact, the assignment matrix-based approach is able to consistently overcome local minima and reach a travel demand estimate that is reasonable. This can be seen in Figure 3a where the 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 3b and Figure 4b, we see that the assignment matrix-based approach can find a reasonable estimate within only a single objective function evaluation, while the assignment matrix-free approach uses a much larger number of evaluations to arrive at reasonable estimates.
Figure 3: The final results from experiments E1-E3. These experiments used the gradient and assignment matrix-based approach.

(a) The ground-truth values plotted against the estimated values.

(b) The objective value history.

Figure 4: The final results from experiments E4-E6. These experiments used the gradient and assignment matrix-free approach.

(a) The ground-truth values plotted against the estimated values.

(b) The objective value history.
The objective value history.

Figure 5: The final results from experiments E7-E9. These experiments used the approach where the k-NN model predict Γ.

(a) The ground-truth values plotted against the estimated values.

(b) The objective value history.

For the machine learning approaches, a feed-forward neural network and k-NN model, the hyperparameters were fine-tuned using a 5-fold cross-validation on 200 sample points. A grid search was performed on the values reported in Table 4. The best performance was achieved by the following settings:

- The feed-forward neural network model with hidden layers containing units \(0.75 \cdot m \cdot n_{G}, 0.25 \cdot m \cdot n_{G}, 0.25 \cdot m \cdot n_{G}\) \((l_{1} = 396, l_{2} = 132, l_{3} = 132)\). This trained model is used in method M4 in Table 2. This model achieved a 5-fold cross-validation score of \(CV_{5} = 589.2922\).
- The k-NN model using \(k = 43\) neighbours. This trained model is used in method M3 in Table 2. This model achieved a 5-fold cross-validation score of \(CV_{5} = 623.0731\).

For the subsequent optimization step using a trained ML model (feed-forward neural network or k-NN) as a surrogate model for DTA, the basinhopping algorithm was used with mostly default parameters. Only the maximum number of iterations was changed and set to 200. Moreover, for the local SLSQP minimizer used as a sub-procedure in the basinhopping algorithm only the default parameters were used.

The estimate obtained from the optimization step is at the end evaluated by SUMO through a DTA as the final 201st objective function evaluation. The final estimate that is evaluated is plotted in Figure 6 for the ML approach that is using a feed-forward neural network and in Figure 5 for the ML approach that is using k-NN.

Across the different experiments on 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 travel demands and the estimated travel demands do not. In other terms, these approaches are able to find travel demand estimates that reproduce the arc counts well but are not at all similar to the ground-truth travel demands.

Disappointingly, the machine learning approaches perform quite poorly: they seem able to improve only slightly during the optimization process, i.e., only a slightly better solution is found than what was already available in the initial data set consisting of 200 sample points.

The reason for this poor performance can possibly be attributed to the goodness-of-fit of the trained machine learning 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, are the high cross-validation scores reported in Section 5.
6 Conclusions and Future Work

The results presented show that the assignment matrix-based approach (method M1) outperforms the assignment matrix-free approach (method M2), as well as the machine learning approaches (method M3-4). Furthermore, based on Table 2 and the plots in Figure 3, 4, 5 and 6, we observe that the assignment matrix-based approach reaches the best estimate with the least computational effort. We also note that even in the case where no seed vectors were supplied, the assignment matrix-based approach could find a reasonable estimate for both the ground-truth travel demands and the corresponding observed arc counts.

A further advantage of the assignment matrix-based approach is that it is easy to apply as no parameters will have to be set or tuned before using the approach. Only 2 new objective function evaluations and thus DTA computations are needed in each iteration, while in comparison, the assignment matrix-free approach needs up to 4. On the other hand, the disadvantage with the assignment matrix-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 machine learning approach in Table 2 and Figures 5 and 6, 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 seem to remain the same independently on which machine learning algorithm is used. Based on these results, the machine learning approach is not good if one seeks a travel demand estimate that is sufficiently close to a given seed vector. That said, it can produce more diverse estimates than, e.g., the assignment matrix-based approach and may be useful for other purposes.

Our conclusion is that the gradient-based approaches work the best. However, the machine learning 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 machine learning approach:

- Investigate further, and in more detail, the sufficient amount of data needed to better fit a model of the SUMO DTA input-output relationship.
- Incorporate more problem-specific structural information about the underlying dynamics in the traffic network that ultimately affect the routing of the users in the network (e.g., as it has been done in [22]).
- Investigate the effect of incorporating additional constraints to further constrain the search space and possibly obtain better estimates.
• Try to incorporate in the models other types of information. For example, there is a possibility to model and predict the assignment matrix (using convolutional neural networks or auto-encoders) instead of the arc counts.
• Use information about the complete paths used by users in the network instead of just the arc counts.

Data Availability

A snapshot of the repository [41] containing the codebase and the data that support the findings of this study are archived and openly available in Zenodo at [42].
References

[1] P. A. Lopez, M. Behrisch, L. Bicker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, “Microscopic traffic simulation using sumo,” IEEE Intelligent Transportation Systems Conference (ITSC), 2018.

[2] T. Abrahamsson, “Estimation of origin-destination matrices using traffic counts - a literature survey,” 1998. (Interim Report) International Institute for Applied Systems Analysis, Laxenburg, Austria.

[3] V. Marzano, A. Papola, and F. Simonelli, “Limits and perspectives of effective o-d matrix correction using traffic counts,” Transportation Research Part C: Emerging Technologies, vol. 17, no. 2, pp. 120 – 132, 2009.

[4] R. Omrani and L. Kattan, “Demand and supply calibration of dynamic traffic assignment models: Past efforts and future challenges,” Transportation Research Record, vol. 2283, no. 1, pp. 100–112, 2012.

[5] T. Djukic, “Dynamic od demand estimation and prediction for dynamic traffic management,” 2014. (PhD thesis) Delft University of Technology, Delft, The Netherlands.

[6] R. Balakrishna, “Off-line calibration of dynamic traffic assignment models,” 2006. (PhD thesis) Massachusetts Institute of Technology, Cambridge, MA, USA.

[7] C. Lindveld, “Dynamic o-d matrix estimation: A behavioural approach,” 2003. (PhD thesis) Delft University of Technology, Delft, The Netherlands.

[8] C. Antoniou, J. Barceló, M. Breen, M. Bullejos, J. Casas, E. Cipriani, B. Ciuffo, T. Djukic, S. Hoogendoorn, V. Marzano, L. Montero, M. Nigro, J. Perarnau, V. Punzo, T. Toledo, and H. van Lint, “Towards a generic benchmarking platform for origin-destination flows estimation/updating algorithms: Design, demonstration and validation,” Transportation Research Part C: Emerging Technologies, vol. 66, pp. 79 – 98, 2016. Advanced Network Traffic Management: From dynamic state estimation to traffic control.

[9] T. Toledo and T. Kolechkina, “Estimation of dynamic origin–destination matrices using linear assignment matrix approximations,” IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 2, pp. 618 – 626, 2013.

[10] R. Frederix, F. Viti, and C. M. Tampère, “Dynamic origin–destination estimation in congested networks: theoretical findings and implications in practice,” Transportmetrica A: Transport Science, vol. 9, no. 6, pp. 494–513, 2013.

[11] T. Djukic, D. Masip, M. Breen, and J. Cascas, “Modified bi-level optimization framework for dynamic od demand estimation in the congested networks,” 39th Australasian Transport Research Forum (ATRF 2017), pp. 1 – 15, 2017.

[12] S. Shaﬁei, M. Saberi, A. Zockaie, and M. Sarvi, “Sensitivity-based linear approximation method to estimate time-dependent origin–destination demand in congested networks,” Transportation Research Record, vol. 2669, no. 1, pp. 72–79, 2017.

[13] D. Masip, T. Djukic, M. Breen, and J. Cascas, “Efﬁcient od matrix estimation based on metamodel for nonlinear assignment function,” 40th Australasian Transport Research Forum (ATRF), pp. 1 – 15, 2018.

[14] J. C. Spall, “Implementation of the simultaneous perturbation algorithm for stochastic optimization,” IEEE Transactions on Aerospace and Electronic Systems, vol. 34, no. 3, pp. 817–823, 1998.

[15] V. Vaze, C. Antoniou, Y. Wen, and M. Ben-Akiva, “Calibration of dynamic traffic assignment models with point-to-point traffic surveillance,” Transportation Research Record, vol. 2090, no. 1, pp. 1–9, 2009.

[16] E. Cipriani, M. Florian, M. Malhot, and M. Nigro, “A gradient approximation approach for adjusting temporal origin–destination matrices,” Transportation Research Part C: Emerging Technologies, vol. 19, no. 2, pp. 270 – 282, 2011. Emerging theories in traffic and transportation and methods for transportation planning and operations.

[17] A. Tympakianaki, H. N. Koutsopoulos, and E. Jenelius, “c-spsa: Cluster-wise simultaneous perturbation stochastic approximation algorithm and its application to dynamic origin–destination matrix estimation,” Transportation Research Part C: Emerging Technologies, vol. 55, pp. 231 – 245, 2015. Engineering and Applied Sciences Optimization (OPT-i) - Professor Matthew G. Karlaftis Memorial Issue.

[18] C. Antoniou, C. L. Azevedo, L. Lu, F. Pereira, and M. Ben-Akiva, “W-spса in practice: Approximation of weight matrices and calibration of trafﬁc simulation models,” Transportation Research Part C: Emerging Technologies, vol. 59, pp. 129 – 146, 2015. Special Issue on International Symposium on Transportation and Traffic Theory.

[19] L. Lu, Y. Xu, C. Antoniou, and M. Ben-Akiva, “An enhanced spsa algorithm for the calibration of dynamic traffic assignment models,” Transportation Research Part C: Emerging Technologies, vol. 51, pp. 149 – 166, 2015.
[20] S. Carrese, E. Cipriani, L. Mannini, and M. Nigro, “Dynamic demand estimation and prediction for traffic urban networks adopting new data sources,” *Transportation Research Part C: Emerging Technologies*, vol. 81, pp. 83 – 98, 2017.

[21] C. Audet and W. Hare, *Derivative-Free and Blackbox Optimization*. Springer International Publishing, 2017.

[22] C. Osorio, “High-dimensional offline origin-destination (od) demand calibration for stochastic traffic simulators of large-scale road networks,” *Transportation Research Part B: Methodological*, vol. 124, pp. 18 – 43, 2019.

[23] L. Kattan and B. Abdulhai, “Noniterative approach to dynamic traffic origin-destination estimation with parallel evolutionary algorithms,” *Transportation Research Record*, vol. 1964, no. 1, pp. 201–210, 2006.

[24] R. Omrani, “Off-line multi-sensor multi-source calibration of dynamic traffic assignment: Simultaneous demand-supply estimation based on genetic algorithms in a high-performance computer,” 2014. (PhD thesis) University of Calgary, Calgary, Canada.

[25] T. Tsakonas, L. Dimitriou, and A. Stathopoulos, “Simultaneous origin-destination matrix estimation in dynamic traffic networks with evolutionary computing,” *Applications of Evolutionary Computing*, pp. 668 – 677, 2007.

[26] K. Ashok and M. Ben-Akiva, “Alternative approaches for real-time estimation and prediction of time-dependent origin-destination flows,” *Transportation Science*, vol. 34, no. 1, pp. 21–36, 2000.

[27] C. Antoniou, R. Balakrishna, H. N. Koutsopoulos, and M. Ben-Akiva, “Off-line and on-line calibration of dynamic traffic assignment systems,” *IFAC Proceedings Volumes*, vol. 42, no. 15, pp. 104 – 111, 2009. 12th IFAC Symposium on Control in Transportation Systems.

[28] T. Friesz and B. D., “Foundation of Network Optimization and Games”. Springer International Publishing, 2016.

[29] S. Peeta and A. K. Ziliaskopoulos, “Foundations of dynamic traffic assignment: The past, the present and the future,” *Networks and Spatial Economics*, vol. 1, no. 3, pp. 233–265, 2001.

[30] C. Gawron, “Simulation-based traffic assignment,” 1998. (PhD thesis) University of Cologne, Cologne, Germany.

[31] J. C. Spall, “Multivariate stochastic approximation using a simultaneous perturbation gradient approximation,” *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.

[32] P. M. Pardalos, A. Zhigljavsky, and J. Žilinskas, *Advances in Stochastic and Deterministic Global Optimization*. Springer International Publishing, Switzerland, 2016.

[33] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, New York, NY, 2009.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[35] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*. Springer-Verlag New York, 2013.

[36] D. J. Wales and J. P. K. Doye, “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*, vol. 101, p. 5111–5116, Jul 1997.

[37] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, I. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.

[38] F. Viti, M. Rinaldi, F. Corman, and C. M. Tampère, “Assessing partial observability in network sensor location problems,” *Transportation Research Part B: Methodological*, vol. 70, pp. 65 – 89, 2014.

[39] S.-R. Hu and H.-T. Liou, “A generalized sensor location model for the estimation of network origin-destination matrices,” *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 93 – 110, 2014.

[40] S.-R. Hu and H.-T. Liou, “A generalized sensor location model for the estimation of network origin-destination matrices,” *Transportation Research Part B: Methodological*, vol. 70, pp. 65 – 89, 2014.

[41] S.-R. Hu and H.-T. Liou, “A generalized sensor location model for the estimation of network origin-destination matrices,” *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 93 – 110, 2014.

[42] S.-R. Hu and H.-T. Liou, “A generalized sensor location model for the estimation of network origin-destination matrices,” *Transportation Research Part B: Methodological*, vol. 70, pp. 65 – 89, 2014.

[43] S.-R. Hu and H.-T. Liou, “A generalized sensor location model for the estimation of network origin-destination matrices,” *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 93 – 110, 2014.