A Unified Framework for End-to-End Learning of User Equilibrium

Zhichen Liu[1] and Yafeng Yin[1]

[1]Department of Civil and Environmental Engineering, University of Michigan, yafeng@umich.edu

Short summary

This paper establishes an end-to-end learning framework for constructing transportation network equilibrium models. The proposed framework directly learns supply and demand components as well as equilibrium states from multi-day traffic state observations. Specifically, it parametrizes unknown model components with neural networks and embeds them in an implicit layer to enforce user equilibrium conditions. By minimizing the differences between the predicted and observed traffic states, parameters for supply and demand components are simultaneously estimated. For efficient training, we design an auto-differentiation-based gradient descent algorithm that handles link- and path-based user equilibrium constraints. The proposed framework is demonstrated using synthesized data on Sioux Falls.
Keywords: Traffic network equilibrium, end-to-end learning, neural networks, and auto-differentiation

1 Introduction

The transportation network equilibrium analysis framework models the interaction between a road network (i.e., supply) and travelers (i.e., demand) and estimates the traffic flow distribution across the network at an equilibrium state. This equilibrium state, referred to as the Wardrop equilibrium or *user equilibrium*, is achieved when no travelers can benefit from unilaterally changing their travel choices (Wardrop, 1952). The user equilibrium then provides a benchmark for designing and comparing potential improvement plans, thereby supporting the planning and management of the transportation network. A static network equilibrium model typically consists of three components that require calibration using empirical data. They are *link performance functions*, which determine travel time based on link flow, *demand functions*, which specify the travel demand between each origin-destination (OD) pair, and *cost or utility functions*, which encapsulate travelers' travel choice preferences. Initiated by Beckmann et al. (1956), this paradigm originally modeled route choices in static, deterministic networks. Since then, it has evolved, now capturing other travel choices such as mode choice, enhanced travel behaviors such as bounded rationality (Xu et al., 2011; Ding et al., 2023), and traffic dynamics (Wang et al., 2018).

All these network equilibrium models have been constructed using a *"bottom-up" assembly approach* where modelers individually pre-calibrate the aforementioned three model components and then assemble them together. To construct an equilibrium model, modelers would start by adopting a particular assumption or theory regarding how travelers make travel choices over a congestible network, and then deduce the corresponding equilibrium conditions. These conditions will be subsequently formulated as an equivalent mathematical program or variational inequality (VI) (alternatively fixed-point or nonlinear complementarity problem) whose solutions prescribe the equilibrium flow distribution.

The bottom-up assembly approach is justified when each model component can be properly determined and calibrated, which, unfortunately, is not the case. For one thing, it is very challenging to properly specify and calibrate a link performance function because congestion does not persist as a steady state in practice. The appropriate specification of a link performance function depends on the underlying dynamics, which is often unobservable to modelers (Small & Chu, 2003). It is also difficult to observe empirical OD demand and even more so to properly calibrate a demand function due to the endogeneity problem (Zhang et al., 2017). Lastly, the travelers' utilities and travel choice preferences are also difficult to observe. Utility functions, chosen based on modelers' judgments and beliefs, may not accurately reflect real-world travel behaviors (Xu et al., 2011; Chen et al., 2016). Adopting different behavior models can lead to different equilibrium conditions and formulations.

The fundamental limitation of the bottom-up assembly approach lies in the disconnection between the specification and calibration of individual components and the ultimate goal of a network equilibrium model: to prescribe an *equilibrium flow distribution* that matches empirical observations as closely as possible, even though real-world systems never truly reach equilibria and observed flows are not in equilibrium states. To address the limitation, we recently delivered a proof-of-concept for an *"end-to-end" framework* that directly learns route choice preferences and equilibrium states from multi-day flow observations (Liu et al., 2023). This study aims to formally establish an end-to-end framework for network equilibrium analysis. As illustrated in Figure 1, the proposed framework approximates unknown supply and demand components with neural networks and then embeds them in an implicit layer that enforces user equilibrium conditions. During forward propagation (indicated by solid arrows), the framework iteratively updates flow distributions via closed-form rules until reaching user equilibrium. During backpropagation (indicated by dashed arrows), it compares the predicted and observed traffic states and adjusts parameters via Auto-Differentiation (AD) to reduce prediction loss/error.

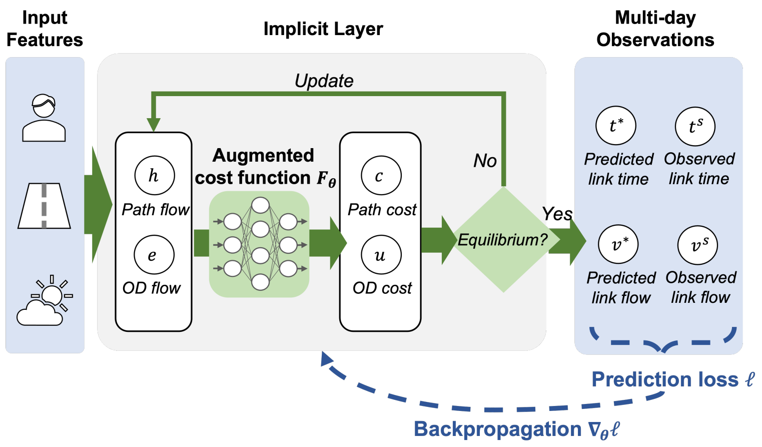

Figure 1: Illustration of the end-to-end framework. This framework combines diverse input features, including traveler characteristics like income, road network attributes like free-flow time, and contextual features like weather. Unknown supply and demand components are integrated into an augmented cost function F_θ, which is parameterized by a neural network with parameters θ.

2 Parametrized User Equilibrium as an Implicit Layer

We consider partial aggregate traffic measures, such as link flow and link time, at peak periods are observable for a long period of time. Supposing a planning agency is interested in developing a static network equilibrium model to analyze the network for peak periods, the end-to-end framework learns the OD demands, travelers' route choice preferences, and link performance functions as well as user equilibrium states from multi-day observations.

Mathematically, consider a network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ and $\mathcal{A}$ are the set of nodes and links. Let $\mathcal{R}$ denote the set of OD pairs. Each OD pair $r \in \mathcal{R}$ is connected by paths that form a finite and nonempty feasible path set $\mathcal{P}_r$ and $\mathcal{P}$ represents the set of feasible paths for all OD pairs. Let $x^{[m]} \in \mathcal{X}$ be the input features observed on day (sample) $m \in \mathcal{M}$. The norm $\|\cdot\|$ denotes Euclidean norm unless otherwise indicated and superscript $[m]$ associates sample-dependent variables with the m-th sample. We focus on using fully connected layers due to their straightforward regularization and analytical properties. Herein, the terms "neural networks" and "fully connected layers" are used interchangeably.

The proposed framework approximates the three components in network equilibrium models with neural networks. We now elaborate on the construction of each component, starting from the supply side. The *link performance function* τ_θ outputs the link time $t^{[m]} \in \mathcal{T}$ as a function of path flow $h^{[m]} \in \mathcal{H}$ and input features $x^{[m]} \in \mathcal{X}$, defined as:

$$\tau_\theta : \mathcal{H} \times \mathcal{X} \to \mathcal{T},$$

where the input features include contextual features (e.g., weather) and road network attributes, such as link capacity and free-flow time; the feasible region $\mathcal{H} \subseteq R_+^{|\mathcal{P}|}$ and $\mathcal{T} \subseteq R_+^{|\mathcal{A}|}$ requires path flow and link time to be nonnegative.

On the demand side, travelers are free to switch paths to improve their utilities. Findings from travel behavior research suggest that route choice behaviors are much more complicated than just choosing the shortest or quickest path. We use the *cost function* π_θ to describe the perceived path cost given actual travel time. The cost function π_θ outputs the (perceived) path cost $c^{[m]} \in \mathcal{C}$ as a continuous function of link time $t^{[m]} \in \mathcal{T}$ and input features, defined as:

$$\pi_\theta : \mathcal{T} \times \mathcal{X} \to \mathcal{C},$$

where input features include traveler characteristics (e.g., income and trip purpose), route attributes (e.g., number of left turns), and contextual features. The feasible set $\mathcal{C} \subseteq R_+^{|\mathcal{P}|}$ requires path cost to be nonnegative. Here, the parameters of all components will be jointly learned and collectively represented as $\theta \in \Theta$.

In addition to route choice, travelers have the freedom to choose whether to travel or not and where to travel to improve their utility. We assume that the OD demands are upper-bounded by maximum possible demands $q \in R_+^{|\mathcal{R}|}$ and introduce the *OD flow* as $e^{[m]} = q - \Gamma^\top h^{[m]}$ to denote the number of trips not realized. Here, $\Gamma \in R^{|\mathcal{P}| \times |\mathcal{R}|}$ represents the path-OD incidence matrix and Γ_{pr} equals 1 if path p connects OD pair r and 0 otherwise. We use an *inverse demand function* λ_θ to depict the OD cost $u^{[m]} \in \mathcal{U}$, which reflects the disutility associated with not traveling, as a function of OD flow $e^{[m]} \in \mathcal{E}$ and input features, namely,

$$\lambda_\theta : \mathcal{E} \times \mathcal{X} \to \mathcal{U},$$

where the feasible region of OD flow is $\mathcal{E} = \{e \in R^{|\mathcal{R}|} : 0 \le e \le q\}$ and $\mathcal{U} \subseteq R_+^{|\mathcal{R}|}$ is the feasible region of OD cost.

Assuming rational travelers try to maximize their travel utilities, the user equilibrium with elastic demand is formulated as the following *parametrized VI*, the solution to which is the equilibrium path flow $h^{*[m]}$ and OD flow $e^{*[m]}$ for sample m:

$$\begin{bmatrix} \pi_\theta(\tau_\theta(h^{*[m]}, x^{[m]}), x^{[m]}) \\ \lambda_\theta(e^{*[m]}, x^{[m]}) \end{bmatrix}^\top \begin{pmatrix} h - h^{*[m]} \\ e - e^{*[m]} \end{pmatrix} \ge 0, \quad \forall h \in \mathcal{H}, e \in \mathcal{E} \tag{1}$$

To simplify notation, we introduce the *augmented flow* as $y = (h, e)$ and the *augmented cost* as $z = (c, u)$. By defining the *augmented cost function* as:

$$F_\theta : \mathcal{Y} \times \mathcal{X} \to \mathcal{Z}$$

where $F_\theta(y, x) = [\pi_\theta^\top(\tau_\theta(h, x), x), \lambda_\theta^\top(e, x)]^\top$, the parametrized VI in Eq.(1) can be compactly reformulated as:

$$\langle F_\theta(y^{*[m]}, x^{[m]}), y - y^{*[m]} \rangle \ge 0, \quad \forall y \in \mathcal{Y}, \tag{2}$$

where the feasible region of augmented flow is $\mathcal{Y} = \{y \in R^{|\mathcal{P}|+|\mathcal{R}|} : y \ge 0, \overline{\Gamma}^\top y = q\}$ if we introduce the augmented path-OD incidence matrix as $\overline{\Gamma} = [\Gamma^\top, I]^\top \in R^{(|\mathcal{P}|+|\mathcal{R}|) \times |\mathcal{R}|}$. If modelers believe the path cost is link-additive, Eq.(1) can also present link-based formulation. The parametrized VI is thereby encapsulated as an implicit layer, which takes contextual features $x^{[m]}$ as input and outputs the augmented equilibrium flow $y^{*[m]}$ such that:

$$y^{*[m]} \in VI\left(F_\theta(y, x^{[m]}), \mathcal{Y}\right).$$

3 End-to-End Learning Framework

We are now ready to discuss how to learn the neural network parameters and user equilibrium flow from multi-day observations. Consider a training dataset $\mathcal{S} = \left\{\left(x^{[m]}, y^{s[m]}\right)\right\}_{m \in \mathcal{M}}$ where

$y^{s[m]}$ is flow observations. Each sample $(x^{[m]}, y^{s[m]})$ is assumed to be drawn independently from an unknown fixed probability distribution $\mathbf{P}$ over $\mathcal{X} \times \mathcal{Y}$. We consider a smooth loss function, denoted as $\ell : \mathcal{Y} \times \mathcal{Y} \to R_+$, to measure the distance between the predicted equilibrium flow $y^{*[m]}$ and corresponding observations. Additionally, modelers can add a regularization term $r(\theta)$ based on their prior knowledge of the model's structure. For instance, they can use $r(\theta) = \|\theta\|$ when expecting sparse parameters. The training process thereby solves the following MPEC:

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{|\mathcal{M}|} \sum_{m=1}^{|\mathcal{M}|} \ell\Big(y^{*[m]}, y^{s[m]}\Big) + r(\theta) \\ s.t. \quad & y^{*[m]} \in VI\left(F_\theta(y, x^{[m]}), \mathcal{Y}\right), \quad \forall m \in \mathcal{M} \end{aligned} \tag{3}$$

The loss function ℓ is flexible to accommodate modelers' needs and available data. It can include partial aggregate traffic state observations like link flow and travel time, path choice probabilities from trajectory data, and benchmark OD demands from planning agencies. The framework integrates multi-source data into a single loss function and effectively reconciles inconsistencies among different data sources.

For scalable computation, we employ an AD-based gradient descent algorithm to solve the MPEC in Eq.(3). To highlight how the training process iteratively updates θ, we will omit input features and denote the augmented cost function $F_\theta(y, x)$ as $F(\theta, y)$ for the remainder of this section and represent the *objective function* as:

$$\Phi(\theta) := f(\theta, y) := \ell(y(\theta), y^s) + r(\theta).$$

Assumption 1 *The augmented cost function $F_\theta(y, x)$ is jointly continuous, μ-strongly monotone and L-Lipshictz continuous in augmented flow y and the feasible region for parameters, augmented flow, and input features are compact.*

Under Assumption 1, the equilibrium state is unique and is a Lipschitzs continuous function of parameter and input feature (Dafermos, 1988). Moreover, the solution to the parametrized VI in Eq.(2) can be formulated as the fixed point of the projection operator, and the MPEC in Eq.(3) becomes:

$$\begin{aligned} \min_{\theta} \quad & \Phi(\theta) := \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} f(\theta, y^{*[m]}(\theta)) \\ s.t. \quad & y^{*[m]}(\theta) = g\left(\theta, y^{*[m]}(\theta)\right), \quad \forall m \in \mathcal{M} \end{aligned} \tag{4}$$

Later numerical examples demonstrate that our analysis remains insightful even when these assumptions are not strictly met.

The training process involves k epochs, each requiring differentiation through the equilibrium flow. To ensure well-posedness, we focus on the differentiable region of the projection operator thereby keeping it within the differentiable programming region for convergence analysis, as stated in the following assumption.

Assumption 2 *The projection operator $g(\theta, y)$ is differentiable in θ and y.*

A similar assumption has been adopted by Li et al. (2022). Addressing the non-differentiability at the boundary of the feasibility set $\mathcal{Y}$ remains an open question.

In each training epoch, we handle two sub-problems: *forward propagation*, which finds an approximate equilibrium flow via N iterations, and *backpropagation*, which uses AD to approximate the gradient of the objective function with respect to the parameter and updates parameters. In the following discussion on forward and backward propagation, variables with a subscript k refer to the k-th epoch. Superscripts n and q denote the n-th forward and q-th backward iteration, respectively. The dependency on sample m is omitted for simplicity.

Forward propagation

The forward propagation updates the augmented flow via N-step projection operation, namely,

$$y_k^n = g(\theta_k, y_k^{n-1}), \quad \forall n \in [N],$$

with step size $\alpha > 0$. In practice, we can extend the projection operator to a general *fixed-point operator*, denoted as $G(\theta, y) : \Theta \times \mathcal{Y} \to \mathcal{Y}$, such that $y^* = \lim_{n\to\infty} G(\theta, y^n)$. Then the general forward propagation follows:

$$y_k^n := G(\theta_k, y_k^{n-1}), \quad \forall n \in [N]. \tag{5}$$

Note that the end-to-end framework is designed for handling large training data and requires batched operations. Instead of solving a single constrained VI, it solves a batch of VIs during forward propagation. Therefore, we need to use *closed-form* fixed-point operators, which can be encoded with computational graphs, to facilitate parallel computation in this step. For example, in path-based formulations, Patwary et al. (2023) used the method of successive averaging (MSA) while Li et al. (2022) used mirror descent. In link-based formulation, Liu et al. (2023) applied the decoupled projection method with $N > 1$. We use mirror descent for path-based formulations and the decoupled projection for link-based formulations, as these methods have demonstrated efficiency in their respective contexts. An alternative approach to solving the fixed-point condition is root-finding methods like Newton's or Quasi-Newton's methods. Such second-order methods can be numerically unstable as demonstrated in our previous work (Liu et al., 2023) so we will limit our discussion to first-order methods in this study.

Backward propagation

In forward propagation, we consider a practical setting where the parametrized VI is solved with N steps and terminated before reaching perfect equilibrium. Consequently, in backpropagation, we need to approximate the gradient of the objective function with respect to the parameter at a non-equilibrium augmented flow, defined as:

Definition 1 (Approximate hypergradient) *The approximate hypergradient at a non-equilibrium augmented flow $\bar{y}$ is defined as:*

$$\widehat{\nabla}\Phi(\theta) := \nabla_\theta f(\theta, \bar{y}) + \widehat{\nabla}\bar{y}(\theta)\, \nabla_y f(\theta, \bar{y}), \tag{6}$$

where the approximate implicit gradient at a non-equilibrium augmented flow $\bar{y}$ is defined as:

$$\widehat{\nabla}\bar{y}(\theta) := \nabla_\theta g(\theta, \bar{y})\, (\boldsymbol{I} - \nabla_y g(\theta, \bar{y}))^{-1}. \tag{7}$$

We consider two AD-based approximations to sidestep the computationally expensive matrix inversion in Eq.(7).

Iterated differentiation (ITD) memorizes the trajectory of N-step forward iterations and directly backpropagates through the equilibrating trajectory. Recall that in the N-th forward iteration, the augmented flow y_k^N depends on θ_k and y_k^{N-1}, namely $y_k^N = g(\theta_k, y_k^{N-1})$. By applying the chain rule, the hypergradient is then approximated by:

$$\widehat{\nabla}\Phi(\theta_k) = \nabla_\theta f(\theta_k, y_k^N) + \Big(\sum_{j=0}^{N-1} \nabla_\theta g(\theta_k, y_k^j) \prod_{i=j+1}^{N-1} \nabla_y g(\theta_k, y_k^i)\Big)\, \nabla_y f(\theta_k, y_k^N). \tag{8}$$

Under **Inexact implicit differentiation (IMD)**, we first define the *auxiliary variable* as:

$$\nu_k^* = \Big(\boldsymbol{I} - \nabla_y g(\theta_k, y_k^N)\Big)^{-1} \nabla_y f(\theta_k, y_k^N).$$

The approximate hypergradient under IMD follows:

$$\widehat{\nabla}\Phi(\theta_k) = \nabla_\theta f(\theta_k, y_k^N(\theta_k)) + \nabla_\theta g(\theta_k, y_k^N(\theta_k))\nu_k^Q. \tag{9}$$

where the auxiliary variable can be recursively approximated using Q-step fixed-point iteration:

$$\nu_k^q = \nu_k^{q-1} + \gamma\left(\nabla_y f(\theta_k, y_k^N) - \Big[I - \nabla_y g(\theta_k, y_k^N)\Big]\nu_k^{q-1}\right), \quad \forall q \in [Q]$$

where step size $\gamma > 0$.

To sum up, by leveraging the hypergradient approximated by ITD or IMD, the parameter at epoch k is updated with learning rate $\beta > 0$ as:

$$\theta_{k+1} = \theta_k - \beta \widehat{\nabla} \Phi(\theta_k).$$

A warm-start strategy, which initializes y_k^0 as the output of the previous training epoch, is adopted to facilitate convergence.

4 Numerical Experiments

Sioux Falls network consists of 76 links, 28 nodes, and 528 OD pairs. We scale the default demand in Stabler (2023) by a factor of three as the maximum possible OD demand q. The ground-truth inverse demand function for OD pair r follows:

$$u_r(e_r, x^{[m]}) = a_u \cdot t_r^0 \cdot x^{[m]} \cdot \exp\Big(b_u \cdot x_r \cdot x^{[m]} \cdot e_r\Big) \tag{10}$$

where $x_r \in [-1, 1]$ represents a normalized OD-specific feature; $x^{[m]} \in [-1, 1]$ is a one-dimensional sample-dependent contextual feature; t_r^0 is the shortest free-flow time between OD pair r; $a_u = 2$ and $b_u = 4$ are functional parameters. We use the standard BPR function as link performance functions and assume travelers only consider travel time when selecting their paths.

In this example, we focus on learning the inverse demand function and assume both link performance and cost functions are given. We consider that multi-day link flows are observable and the loss function measures the mean square error (MSE) between predicted and observed link flow. The framework is trained using the Adam optimizer over $K = 500$ epochs with early stopping implemented if there is no improvement in the training MSE over 50 consecutive epochs. The forward propagation uses mirror descent with $N = 100$ iterations, while backpropagation uses the ITD method. We partition our data into training, validation, and test sets, containing 1,024, 256, and 256 samples each. To ensure robust results, we run each experiment 10 times and report the average and standard deviation of metrics.

We consider the following four scenarios and fine-tune the learning rate and step sizes via grid search for each setting.

- *Benchmark*: Modelers use grid search to identify a fixed OD demand that best matches all testing samples, which is 1.04 in this case.
- *Functional*: Modelers encode the functional form in Eq.(10) with computational graphs and learns two parameters a_u and b_u.
- *Linear*: Modelers encode the inverse demand function as a linear combination of x_r, $x^{[m]}$, and e_r.
- *End-to-end*: The neural network follows the "kernel strategy" from Liu et al. (2023) to handle potential changes in the number of OD pairs during "what-if" analysis: its input dimension depends on the number of input features, in this case, x_r, $x^{[m]}$, and e_r, rather than the number of ODs. It combines a linear part (as in the *Linear* model) and a nonlinear part, comprising three layers with eight neurons for each middle layer.

Here *Functional* and *Linear* models follow the traditional "bottom-up" approach and preselect the functional form before calibration. Specifically, the *Functional* model assumes modelers have perfect information of the underlying functional form, while the *Linear* model considers the potential inaccuracies in the modelers' understanding when defining this form. On the other hand, the *End-to-end* model adopts an "end-to-end" modeling approach, and neural networks are regularized to ensure monotonicity and Lipschitz continuity. Interested readers can refer to our previous work for more details (Liu et al., 2023). We evaluate the framework performance with *weighted mean absolute percentage error (WMAPE)* φ_v, which quantifies percentage errors in traffic state predictions.

Table 1 presents the WMAPE under different settings. In *Benchmark* model, the link flow WMAPE is remarkably high at 50.5%. This error drops to 0.8% when we encode the ground-truth functional form in the framework and adjust a_u and b_u. The non-zero error can be attributed to the nonconvexity of MPEC, which can trap the training process at a local minimum. The *End-to-end* model has no information about the functional form of the inverse demand function. By leveraging the expressivity of neural networks, it achieves a WMAPE of 1.8% comparable to the *Functional* model. By contrast, the *Linear* model, which follows a "bottom-up" modeling approach, makes a wrong assumption about the functional form and only reduces flow WMAPE to 9.5%. This result supports that the end-to-end framework can autonomously "discover" the equilibrium flow from observations without requiring the knowledge of the model component's functional form.

Model	# Params	Link flow	Link time	Demand
Benchmark	1	50.5 (0.02)	98.0 (0.01)	83.41 (0.06)
Functional	2	0.8 (0.02)	2.2 (0.1)	0.5 (0.01)
Linear	4	9.5 (0.31)	32.4 (2.5)	5.7 (0.24)
End-to-End	126	1.8 (0.29)	4.4 (0.9)	1.3 (0.2)

Table 1: WMAPEs across training settings. Average WMAPEs are shown as percentages with the standard deviation in parentheses.

Next, we investigate the effects of forward iterations and backward methods on framework performance. Figure 2 illustrates the MSE over 70 epochs for two different backward methods. AD-ITD completes 70 epochs faster than IMD when $N = 1$, but takes longer as N increases to 50. This increased computation time is attributed to that a larger N under AD-ITD requires more iterations for both forward and backward propagation (see Figure 2a). By contrast, AD-IMD avoids differentiating along the equilibrating trajectory and the computation time changes relatively mildly when N varies (see Figure 2b). Moreover, increasing N from 1 to 50 reduces training MSE and speeds up convergence regardless of backward methods.

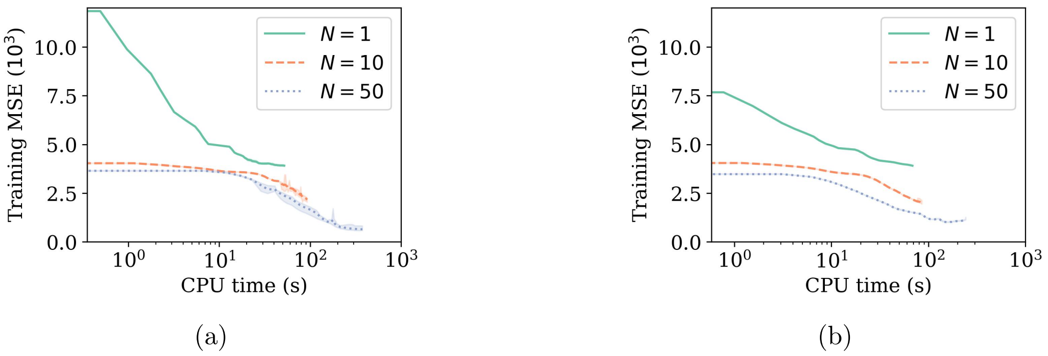

Figure 2: Training MSEs with different forward iterations under (a) ITD and (b) IMD for 70 epochs. Shaded areas in this and the following graphs indicate the standard deviation unless stated otherwise.

Figure 3 shows that an iterative equilibrating process is important to ensure local convergence. As shown in Figure 3a, the training process under AD-ITD stops prematurely with $N = 1$, resulting in a high training MSE around 4×10^3. By contrast, AD-IMD uses extra information from the implicit function theorem to correct AD and keeps reducing the training MSE with $N = 1$. Both AD-ITD and AD-IMD manage to avoid getting stuck when N increases to 10 (see Figure 3b) and AD-ITD outperforms AD-IMD in finding better local optima when N increases to 50 (see Figure 3c).

5 Conclusions

This study establishes an end-to-end network equilibrium model tha directly learns unknown components and user equilibrium states from multi-day observations. To enhance training efficiency, we leverage two AD-based gradient descent algorithms, AD-ITD and AD-IMD, which are capable

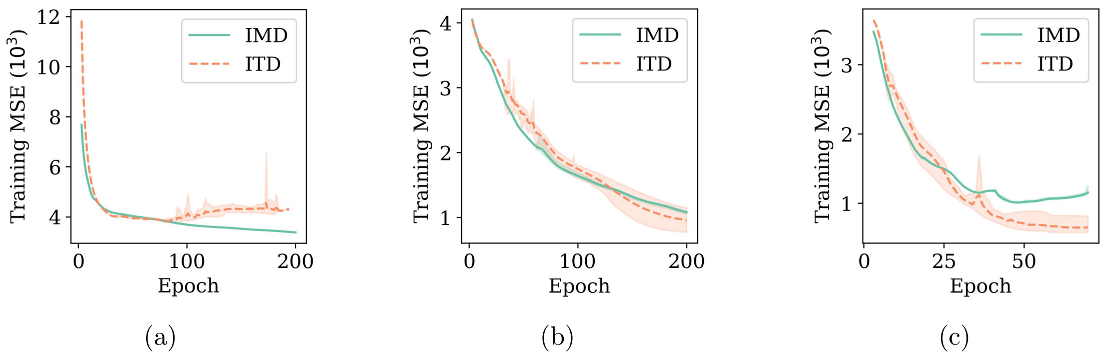

Figure 3: Training MSEs using different backward methods with (a) $N = 1$, (b) $N = 10$, and (c) $N = 50$.

of handling both link- and path-based user equilibrium constraints. Our findings are validated using synthesized data from Sioux Falls networks. Looking ahead, based on our established learning framework, we plan to investigate end-to-end optimization to prescribe potential improvement strategies, such as modifying lane configurations, expanding capacity, and implementing congestion pricing. How to tackle the discrete decision variables – such as the addition of new roads or lanes – will be an area of focus in future studies.

Acknowledgements

The work described in this paper was partly supported by research grants from the USDOT Center for Connected and Automated Transportation, and National Science Foundation CMMI- 2233057.).

References

Beckmann, M., McGuire, C. B., & Winsten, C. B. (1956). *Studies in the economics of transportation* (Tech. Rep.).

Chen, C., Ma, J., Susilo, Y., Liu, Y., & Wang, M. (2016). The promises of big data and small data for travel behavior (aka human mobility) analysis. *Transportation research part C: emerging technologies*, *68*, 285–299.

Dafermos, S. (1988). Sensitivity analysis in variational inequalities. *Mathematics of Operations Research*, *13*(3), 421–434.

Ding, H., Yang, H., Xu, H., & Li, T. (2023). Status quo-dependent user equilibrium model with adaptive value of time. *Transportation Research Part B: Methodological*, *170*, 77–90.

Li, J., Yu, J., Wang, Q., Liu, B., Wang, Z., & Nie, Y. M. (2022). Differentiable bilevel programming for stackelberg congestion games. *arXiv preprint arXiv:2209.07618*.

Liu, Z., Yin, Y., Bai, F., & Grimm, D. K. (2023). End-to-end learning of user equilibrium with implicit neural networks. *Transportation Research Part C: Emerging Technologies*, *150*, 104085.

Patwary, A., Wang, S., & Lo, H. K. (2023). Iterative backpropagation method for efficient gradient estimation in bilevel network equilibrium optimization problems. *Transportation Science*.

Small, K. A., & Chu, X. (2003). Hypercongestion. *Journal of Transport Economics and Policy (JTEP)*, *37*(3), 319–352.

Stabler, B. (2023). *Transportationnetworks.* `https://github.com/bstabler/TransportationNetworks`.

Wang, Y., Szeto, W. Y., Han, K., & Friesz, T. L. (2018). Dynamic traffic assignment: A review of the methodological advances for environmentally sustainable road transportation applications. *Transportation Research Part B: Methodological*, *111*, 370–394.

Wardrop, J. G. (1952). Road paper. some theoretical aspects of road traffic research. *Proceedings of the institution of civil engineers*, *1*(3), 325–362.

Xu, H., Lou, Y., Yin, Y., & Zhou, J. (2011). A prospect-based user equilibrium model with endogenous reference points and its application in congestion pricing. *Transportation Research Part B: Methodological*, *45*(2), 311–328.

Zhang, C., Osorio, C., & Flötteröd, G. (2017). Efficient calibration techniques for large-scale traffic simulators. *Transportation Research Part B: Methodological*, *97*, 214–239.