LEARNING OPTIMAL TRAFFIC ROUTING BEHAVIORS USING MARKOVIAN FRAMEWORK IN MICROSCOPIC SIMULATION

Theophile Cabannes
University of California, Berkeley
LIX, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris
652 Sutardja Dai Hall, Berkeley, CA 94720-1710
email: theophile@berkeley.edu

Jiayi Li
University of California, Berkeley
652 Sutardja Dai Hall, Berkeley, CA 94720-1710
email: lijiayi9712@berkeley.edu

Fangyu Wu
University of California, Berkeley
652 Sutardja Dai Hall, Berkeley, CA 94720-1710
email: fangyuwu@berkeley.edu

Harry Dong
University of California, Berkeley
652 Sutardja Dai Hall, Berkeley, CA 94720-1710
email: hdong920@berkeley.edu

Alexandre M.Bayen
University of California, Berkeley
Institute of Transportation Studies
109 McLaughlin Hall, Berkeley CA 94720-1720
email: bayen@berkeley.edu

Submission Date: July 31, 2019

ABSTRACT

This article applies the existing Markovian traffic assignment framework to novel traffic control strategies. In the Markovian traffic assignment framework, transition matrices are used to derive the traffic flow allocation. In contrast to the static traffic assignment, the framework only requires flow split ratio at every intersection, bypassing the need of computing path flow allocation. Consequently, compared to static traffic assignment, drivers' routing behaviors can be modeled with fewer variables. As a result, it could be used to improve the efficiency of traffic management, especially in large scale applications. To begin with, the article introduces Markovian traffic assignment and connects it to the classic static traffic assignment. Then, the framework is extended to dynamic traffic assignment using microscopic traffic simulator *Simulation of Urban Mobility* (SUMO). In a case study, the framework is applied to a standard benchmark network, where optimal routing behaviors are independently learned through grid search, random search, and evolution strategies, under three different reward functions (network outflow, total vehicle hours of travel, and average marginal regret). The case study shows that the this novel traffic control strategy is promising, as Markov chain theory supports the ability to scale up to larger networks.

Keywords: Network equilibrium modeling, Routing behavior, Nash games, Markov chains

INTRODUCTION

Motivations

Reducing traffic congestion.

In many parts of the world, congestion has swelled to be a difficult dilemma (*1*). In 2018, INRIX calculated an annual loss of around $305 billion in the U.S. due to the direct and indirect impacts of congestion on productivity (*2*). Left unattended, congestion can amplify the complications that currently come along with it, which include but are not limited to wasted time, excessive noise, unnecessary fuel consumption, and increased greenhouse gas emissions (*3*). Researchers and practitioners have tried many ways to reduce fuel consumption and greenhouse gas emissions by improving the quality of vehicles themselves, but this only makes congestion more bearable without solving the problem itself (*3*).

The need of models to perform traffic estimation using new data sources.

With the rise of smart phones and cloud computing, traditional traffic management has been outpaced by new mobility services, such as transportation network companies and navigational apps (*4*). Consequently, current research aims to improve traffic models by leveraging new mobility services, such as real-time, GPS-based, point-speed data from mobile devices (*5*). Even so, the data sources required for current traffic models are challenging to collect, as current traffic models need trajectory data, yet traffic data are mainly available as cross-sectional data. Therefore, this article focuses on developing Markov chain-based models to utilize link flow data more effectively.

Opportunities to improve traffic flow control through network connectivity.

Real-time traffic predictions can help reduce congestion by improving traffic control strategies (*6*). Additionally, traffic control systems could benefit from the use of navigational apps and autonomous vehicles (*7*). However, controlling every vehicle can be expensive and may lead to ethical issues. Therefore, this article considers network traffic control strategies based on controlling vehicle split ratios at every intersection. Such a decentralized structure allows for massive parallelization: at every intersection, traffic control can be applied through infrastructure-to-vehicle communications independent of other intersections, which is a significant computational advantage.

Contributions

Markov chains to model traffic assignment

In this article, Markov chains are used to model network traffic equilibrium flow. This is claimed to be similar to Google's use of Markov chains to model internet congestion in (*8*). In the Markovian traffic assignment framework, transition matrices are used to derive traffic flow allocations. More specifically, for any vehicle arriving at an intersection, the transition matrix indicates its probability to go on any downstream road segments from its current road segment. Using the law of large numbers, each element of the transition matrix can be seen as the flow split ratio of every intersection. Therefore, Markov chains can extend the static traffic assignment: the Markovian traffic assignment framework.

Key contributions

The key contributions of the article include:

- Framing the static traffic assignment into an equivalent Markov chain framework. The clas-

sic static traffic assignment (STA) is reframed into an optimization problem on a transition matrix instead of path flows. At each intersection, incoming flows are split among the downstream roads according to the split ratios given by the transition matrix of the network. This reformulation can be seen as an Eulerian formulation of a Lagrangian problem using fluid mechanics concepts (*9*).

- Extending the Markov chain to the dynamic traffic assignment. The Markovian traffic framework is extended to the dynamic traffic assignment using the microscopic simulator SUMO. A benchmark network composed of five links connected by two branches and one merge is set up in SUMO. Every vehicle path is generated by following the probabilities given by the transition matrix of the Markovian framework.
- Learning optimal routing behavior with the dynamic Markovian framework. We demonstrate how to optimize the traffic flow allocation under the Markovian framework in a case study. To begin, three reward functions – network outflow, total vehicle hours of travel, and average marginal regret – are designed to measure the effectiveness of traffic flow allocations. Next, grid search, random search, and evolution strategies are performed on the transition matrix to optimize routing behaviors based on the simulation data collected from SUMO. The results show that those three methods are comparable in the case study under all three rewards.

Outline

This article first explains the classic traffic assignment (*10*) and introduces the Markovian static traffic assignment in Section "Traffic Assignment Dynamics". Second, in order to compare both models, optimal flow allocations are defined according to the notions of user equilibrium and social optimum in Section "Optimal Static Flow Allocation". Next, the Markovian framework is extended to dynamic traffic assignment with the help of SUMO to compute the flow allocations using transition matrices in Section "Dynamic Traffic Assignment." In Section "Learning Optimal Flow Allocation", we search the decision space of constructed dynamic traffic assignment scenario through grid search, random search, and evolution strategies.

TRAFFIC ASSIGNMENT DYNAMICS

This section introduces the static traffic assignment framework (*10*) and an equivalent Markovian framework (*6*). In the Markovian framework, computation of the optimal link flow allocation is done using transition matrices. This bypasses the need of calculating path flows in the static traffic assignment which can be computationally expensive.

Notations

This section defines a network, or graph, $\mathscr{G}$ which is composed of nodes, or vertices, $\mathscr{N}$ joined by links, or edges, $\mathscr{L}$. The network is assumed to be strongly connected, and the set of paths without cycles is defined as $\mathscr{P}$. For each path $\mathbf{p}$, the flow on the path is denoted as $h_{\mathbf{p}}$. For each link l, the flow on the link is denoted as f_l. Given a demand matrix D, which indicates the flows that enter and leave the network at each node, we denote the set of feasible path flow allocations as $\mathscr{H}_D$. The following notations and definitions are derived from (*10*).

Definition 3.1 (Network, paths, and demand)**.** *Given a finite strongly connected directed graph $\mathscr{G}$ with nodes set $\mathscr{N}$ and links set $\mathscr{L} \subset \mathscr{N} \times \mathscr{N}$, i.e. $\mathscr{G} = (\mathscr{N}, \mathscr{L})$, for each origin $o \in \mathscr{N}$ and destination $d \in \mathscr{N}$:*

- *Let $\mathscr{P}_{od}$ be the set of feasible paths without cycles from o to d.*
- *Let $d_{od} \geq 0$ be the total number of vehicles that make the journey $o \to d$, per unit of time. We denote the demand matrix $D = (d_{od})_{o,d \in \mathscr{N}}$.*

For all of the following definitions and sections, $\mathscr{G}$ is assumed to be given and strongly connected. Therefore, for the sake of redundancy, this fact will not be restated every time.

Remark 3.1 (Tails and heads of links)**.** *For any link $l \in \mathscr{L} \subset \mathscr{N} \times \mathscr{N}$, because $\mathscr{L} \subset \mathscr{N} \times \mathscr{N}$, there exists unique $t, h \in \mathscr{N}$ such that $l = (t,h)$. We refer to t as the tail of l, and to h as the head of l.*

We will use the diamond network as a benchmark network to illustrate definitions and notations. Assume that node A is the origin, node D is the destination, and let demand be d_{AD}. This network is shown in Figure 1.

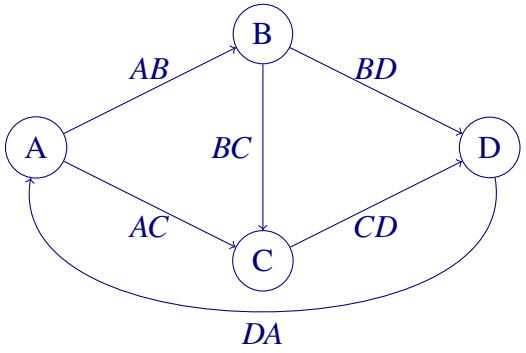


FIGURE 1: Benchmark network to illustrate definitions and notations.

Using the network from Figure 1, we have:

$$\mathscr{N} = \{A,\ B,\ C,\ D\}$$
$$\mathscr{L} = \{DA,\ AB,\ AC,\ BC,\ BD,\ CD\} \qquad \text{where } DA = (D,A),\ AB = \dots$$
$$\mathscr{P}_{AD} = \{ABD,\ ABCD,\ ACD\} \qquad \text{where } ABD = (AB,BD),\ ABCD = \dots$$
$$D = \begin{bmatrix} 0 & 0 & 0 & d_{AD} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Definition 3.2 (Path flows, feasible path flow allocation)**.** *For each path $\mathbf{p} \in \mathscr{P} = \bigcup_{o,d \in \mathscr{N}} \mathscr{P}_{od}$:*

- *let $h_{\mathbf{p}}$ be the path flow of path $\mathbf{p}$: the number of vehicles using path $\mathbf{p}$ per unit of time. We denote the path flow vector $\mathbf{h} = (h_{\mathbf{p}})_{\mathbf{p} \in \mathscr{P}}$.*

Given a demand matrix $D \in \mathbb{R}_+^{|\mathscr{N}| \times |\mathscr{N}|}$:

- *let $\mathscr{H}_D = \left\{ \mathbf{h},\ \mathbf{p} \in \mathscr{P} : h_{\mathbf{p}} \in \mathbb{R}_+,\ o,d \in \mathscr{N} : \sum_{\mathbf{p} \in \mathscr{P}_{od}} h_{\mathbf{p}} = d_{od} \right\}$ be the set of feasible path flow allocations.*

In Figure 1, a path flow allocation $\mathbf{h} = (h_\mathbf{p})_{\mathbf{p}\in\mathscr{P}}$ is feasible if and only if: $h_{ABD} + h_{ABCD} + h_{ACD} = d_{AD}$.

Definition 3.3 (Link flows)**.**

- *For each link $l \in \mathscr{L}$, let f_l be the link flow of l: the number of vehicles using link l per unit of time.*
- *We denote the link flow vector $\mathbf{f} = (f_l)_{l\in\mathscr{L}}$.*

Static traffic assignment notation

In this section, we define the static traffic assignment assumption that will be relaxed in the section "Dynamic traffic assignment using SUMO". Following definitions from (*10*), link travel times t_l and path travel times $c_\mathbf{p}$ are defined. Assuming stationary conditions (which are reasonable for periods of time where demand is quasi-static), the link flow allocation $\mathbf{f}$ can be derived from the path flow allocation $h_\mathbf{p}$ using the incidence matrix Δ: $\mathbf{f} = \Delta\mathbf{h}$. Similarly, $c_\mathbf{p} = \Delta^\top \mathbf{t}$.

Definition 3.4 (Incidence matrix)**.**

- *For each path $\mathbf{p} \in \mathscr{P}$ and link $l \in \mathscr{L}$, we define* $\delta_{l,\mathbf{p}} = \begin{cases} 1 & \text{if } l \in \mathbf{p} \\ 0 & \text{otherwise} \end{cases}$.
- *We denote $\delta_\mathbf{p} = (\delta_{l,\mathbf{p}})_{l\in\mathscr{L}}$ the indicator vector of the links included in path $\mathbf{p}$.*
- *We denote $\Delta = (\delta_{l,\mathbf{p}})_{l\in\mathscr{L},\mathbf{p}\in\mathscr{P}}$ the incidence matrix.*

On the benchmark network of Figure 1, considering $\mathscr{P} = \mathscr{P}_{AD}$, we have:

$$\Delta = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Definition 3.5 (Travel time)**.**

- *For each link $l \in \mathscr{L}$, let t_l be the travel time on link l.*
- *We denote the link travel time vector $\mathbf{t} = (t_l)_{l\in\mathscr{L}}$.*
- *For each path $\mathbf{p} \in \mathscr{P}$, we define $c_\mathbf{p}$ as the travel time on path $\mathbf{p}$ as the sum of the travel times on each link that is included in path $\mathbf{p}$, i.e. $c_\mathbf{p} = \delta_\mathbf{p}^\intercal \cdot \mathbf{t}$.*
- *We denote the path travel time vector $\mathbf{c} = (c_\mathbf{p})_{\mathbf{p}\in\mathscr{P}}$.*

Definition 3.6 (Static model, feasible link flow allocation)**.** *Static equilibrium conditions are assumed. Therefore, for any path $\mathbf{p}$, $h_\mathbf{p}$ remains constant over time and $\mathbf{f} = \Delta\mathbf{h}$.*

Given a demand matrix $D \in \mathbb{R}_+^{|\mathscr{N}|\times|\mathscr{N}|}$, let $\mathscr{F}_D = \{\Delta\mathbf{h},\ \mathbf{h} \in \mathscr{H}_D\}$ be the set of feasible link flow allocations.

Definition 3.7 (Separability of travel time, travel time function)**.** *For each $l \in \mathscr{L}$, t_l is (only) a function of f_l: $t_l(f_l)$. We assume that for each $l \in \mathscr{L}$, t_l is a strictly increasing function, and is positive for $f_l \geq 0$. We denote $\mathbf{t}(\mathbf{f}) = (t_l(f_l))_{l\in\mathscr{L}}$.*

For every link flow allocation $\mathbf{f} \in \mathscr{F}_D$, the travel time of each path $\mathbf{p} \in \mathscr{P}$ is $c_\mathbf{p}(\mathbf{f}) = \delta_\mathbf{p}^\intercal \cdot \mathbf{t}(\mathbf{f})$.

Markovian framework for static traffic assignment

In this section, the Markovian framework, inspired by (*6, 11*), is presented. First, the line graph $L(\mathscr{G})$ is defined as the graph where the roles of nodes and links in $\mathscr{G}$ are swapped. Next, based on the adjacency matrix of $L(\mathscr{G})$, the transition matrix P – a type of stochastic matrix (*12*) – gives the transition probabilities to go from any link l to its downstream links. Equilibrium flows (in the context of Markov chains) are then defined using the transition matrix P and the demand matrix D.

Definitions

Definition 3.8 (Line graph (*13*))**.** *The line graph $L(\mathscr{G}) = (\mathscr{N}_{L(\mathscr{G})}, \mathscr{L}_{L(\mathscr{G})})$ is the graph formed from swapping the roles of nodes and links in $\mathscr{G} = (\mathscr{N}, \mathscr{L})$. Each link in $\mathscr{G}$ is represented as a node in $L(\mathscr{G})$: $\mathscr{N}_{L(\mathscr{G})} = \mathscr{L}$, and each pair of links (l_1, l_2) in $\mathscr{G}$ such that the head of l_1 is the tail of l_2 is a link in $L(\mathscr{G})$.*

Remark 3.2 (Line graphs vs. dual graphs)**.** *Such a graph is called a line graph, but some articles – such as (6) and (14) – use the term dual graph instead. Note that the definition of the dual graph (in (6) and (14)) is not equivalent to the definition of dual graphs from a graph theory point of view, where the faces in the original graphs are nodes in the dual graph.*

Definition 3.9 (Stochastic matrix (*12*))**.** *For a line graph $L(\mathscr{G})$, $S \in \mathbb{R}^{|\mathscr{L}|\times|\mathscr{L}|}$ is defined as a stochastic matrix if and only if:*

$$\begin{aligned}
&\forall i, j \in \mathscr{L}, && s_{i,j} \geq 0 \\
&S\mathbf{1} = \mathbf{1} && \\
&\forall i, j \in \mathscr{L}, && s_{i,j} \neq 0 \implies (i,j) \in \mathscr{L}_{L(\mathscr{G})}
\end{aligned}$$

Definition 3.10 (Transition matrix (*8*))**.** *For a line graph $L(\mathscr{G})$, $P \in \mathbb{R}^{|\mathscr{L}|\times|\mathscr{L}|}$ is defined as a transition matrix if and only if:*

$$\begin{aligned}
&P \text{ is a stochastic matrix} \\
&\forall i_1, i_2, l \in \mathscr{L}, \quad p_{i_1,l} \neq 0 \text{ and } p_{i_2,l} \neq 0 \implies p_{i_1,l} = p_{i_2,l}
\end{aligned}$$

We denote $\mathscr{T}$ as the set of transition matrices.

Remark 3.3 (Line graphs and Markov chains)**.** *The line graph of the network is used because information about flow lies in the road segments (or links of $\mathscr{G}$), not in the actual locations or intersections (or nodes of $\mathscr{G}$). With this setup, well-studied Markov chain properties can be used (12) to observe how flow shifts from one road to the next at every timestep.*

While stochastic matrices allow any probability distribution from one link to the next, transition matrices assume that if two links l_1, l_2 share the same head, then the probability to go from l_1 to a downstream link l_f is the same as the probability to go from l_2 to the same link l_f.

Remark 3.4 (Markov chain)**.** *A Markov chain is a stochastic process in which the probability of changing to a state j from state i depends only on the fact that the current state is i (8). Markov chain theory describes that the transition matrix can be used to compute the expected distribution of flows $\mathbf{f}_{t+1}$ at time $t+1$ from the distribution of flows $\mathbf{f}_t$ at time t: $\mathbf{f}_{t+1} = P^\top \mathbf{f}_t$ (8). Therefore, transition probabilities can also be seen as split ratios (see definition 3.13).*

On the benchmark network (Figure 1), we have that:

$$\mathscr{N}_{L(\mathscr{G})} = \{DA, AB, AC, BC, BD, CD\} = \mathscr{L}$$
$$\mathscr{L}_{L(\mathscr{G})} = \{(DA,AB),(DA,AC),(AB,BD),(AB,BC),(BC,CD),(AC,CD),(BD,DA),(CD,DA)\}$$
$$P = \begin{bmatrix} 0 & p_{AB} & p_{AC} & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{BC} & p_{BD} & 0 \\ 0 & 0 & 0 & 0 & 0 & p_{CD} \\ 0 & 0 & 0 & 0 & 0 & p_{CD} \\ p_{DA} & 0 & 0 & 0 & 0 & 0 \\ p_{DA} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The line graph of the benchmark graph and the corresponding transition matrix are shown in Figure 2 where nodes are in the approximate positions of the links in the benchmark graph. In this case, demand enters at nodes AB and AC.

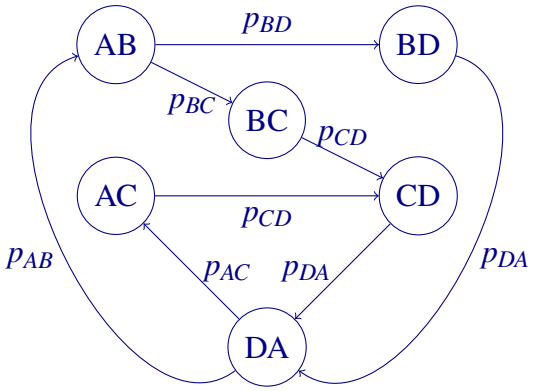


FIGURE 2: The line graph of the benchmark network.

Static equilibrium using Markov chains

Flow movement through networks is described using previously defined line graphs and transition matrices. Transition probabilities, or split ratios, are defined as elements of transition matrices. Then, using the concept of flow conservation, equilibrium flow can be defined using ideas analogous to limits of Markov chains. This is useful later to reframe the current static traffic assignment framework using transition matrices.

Definition 3.11 (Transition probability)**.** *For a transition matrix $P \in \mathscr{T}$, we define $p_{i,l}$ for every link $i \in \mathscr{L}$ and $l \in \mathscr{L}$ as the transition probability to go from the link i to the link l.*

Note that, if there is no connection between the two links (i.e. $(i,l) \in \mathscr{L}_{L(\mathscr{G})}$), then $p_{il} = 0$.

Because, P is a transition matrix, $\forall i_1, i_2, l \in \mathscr{L}$, such that $p_{i_1,l} \neq 0$ and $p_{i_2,l} \neq 0$ implies $p_{i_1,l} = p_{i_2,l}$ (definition), and $\exists i \in \mathscr{L}$ such that $p_{i,l} \neq 0$ ($P\mathbf{1} = \mathbf{1}$), we can uniquely define $p_l = p_{i,l}$ (c.f. remark 3.3).

We call p_l the transition probability of the link l.

Node dynamics

At a specific node n of the network $\mathscr{G}$, the conservation flow inside the network tells that the total inflow of vehicles arriving at the node n added with the demand that arrives at the node is equal to the total outflow of vehicles departing from the node n added with the demand that exits at the node.

This can be summarized into one equality:

Definition 3.12 (Flow conservation)**.** *Given a demand $D \in \mathbb{R}_+^{|\mathscr{N}|\times|\mathscr{N}|}$, a link flow allocation* $\mathbf{f}$ *satisfies the flow conservation if for every node $n \in \mathscr{N}$:*

$$\sum_{u,(u,n)\in\mathscr{L}} f_{u,n} + \sum_{\tilde{u}\in\mathscr{N}} d_{n,\tilde{u}} = \sum_{v,(n,v)\in\mathscr{L}} f_{n,v} + \sum_{\tilde{v}\in\mathscr{N}} d_{\tilde{v},n}$$

Property 3.1 (Flow conservation)**.** *Given a demand $D \in \mathbb{R}_+^{|\mathscr{N}|\times|\mathscr{N}|}$, if a link flow allocation* $\mathbf{f}$ *is feasible (i.e. $\mathbf{f} \in \mathscr{F}_D$) then* $\mathbf{f}$ *satisfies the flow conservation.*

Proof. The proof can be done by showing that:

$$\sum_{u,(u,n)\in\mathscr{L}} \sum_{v,(n,v)\in\mathscr{L}} \sum_{\mathbf{p}\in\mathscr{P}} h_{\mathbf{p}}\delta_{\mathbf{p},(u,n)}\delta_{\mathbf{p},(n,v)} = \sum_{u,(u,n)\in\mathscr{L}} f_{u,n} - \sum_{\tilde{v}\in\mathscr{N}} d_{\tilde{v},n}$$

and

$$= \sum_{v,(n,v)\in\mathscr{L}} f_{n,v} - \sum_{\tilde{u}\in\mathscr{N}} d_{n,\tilde{u}}$$

which shows that

$$\sum_{u,(u,n)\in\mathscr{L}} f_{u,n} + \sum_{\tilde{u}\in\mathscr{N}} d_{n,\tilde{u}} = \sum_{v,(n,v)\in\mathscr{L}} f_{n,v} + \sum_{\tilde{v}\in\mathscr{N}} d_{\tilde{v},n}$$

On the benchmark network example (Figure 1), for the node B, we have:

$$\begin{aligned}\sum_{u,(u,B)\in\mathscr{L}} \sum_{v,(B,v)\in\mathscr{L}} \sum_{\mathbf{p}\in\mathscr{P}} h_{\mathbf{p}}\delta_{\mathbf{p},(u,B)}\delta_{\mathbf{p},(B,v)} &= \delta_{AB,ABD}\delta_{BC,ABD}h_{ABD} + \delta_{AB,ABD}\delta_{BD,ABD}h_{ABD} \\ &\quad + \delta_{AB,ABCD}\delta_{BC,ABCD}h_{ABCD} + \delta_{AB,ABCD}\delta_{BD,ABCD}h_{ABCD} \\ &\quad + \delta_{AB,ACD}\delta_{BC,ACD}h_{ACD} + \delta_{AB,ACD}\delta_{BD,ACD}h_{ACD} \\ &= \delta_{AB,ABD}h_{ABD} + \delta_{AB,ABCD}h_{ABCD} = f_{AB} \\ &= \delta_{BC,ABD}h_{ABD} + \delta_{BC,ABCD}h_{ABCD} + \delta_{BD,ABD}h_{ABD} + \delta_{BD,ABCD}h_{ABCD} = f_{BC} + f_{BD}\end{aligned}$$

So this gives that $f_{AB} = f_{BC} + f_{BD}$.

Equilibrium flow

Using the flow conservation property and transition probabilities, equilibrium flow allocations can now be defined. In fact, it can be shown that the equilibrium flow allocation is a solution to a linear equation involving the transition matrix.

Definition 3.13 (Transition dynamics)**.** *Given a transition matrix $P \in \mathscr{T}$ and demand $D \in \mathbb{R}_+^{|\mathscr{N}|\times|\mathscr{N}|}$, a link flow allocation $\mathbf{f} \in \mathbb{R}^{|\mathscr{L}|}$ is defined as an equilibrium flow allocation if:*

1. $\mathbf{f}$ *satisfies the flow conservation (definition 3.12)*

2. *For every node* $n \in \mathcal{N}$ *and every link* $l \in \mathcal{L}$ *that has tail* n ($\exists m \in \mathcal{N}$ *such that* $l = nm$), $f_l = p_l f_{n,out}$, *with* $f_{n,out} = \sum_{w,(n,w)\in\mathcal{L}} f_{n,w}$
3. $\mathbf{f} \geq 0$

Notation 3.1 (Set of equilibrium flow)**.** *Given a transition matrix* $P \in \mathcal{T}$ *and demand* $D \in \mathbb{R}_+^{|\mathcal{N}|\times|\mathcal{N}|}$, $\mathcal{F}_D(P)$ *is denoted as the set of equilibrium flows.*

Notation 3.2 (Origin probability P_o)**.** *Given* $P \in \mathcal{T}$, *we denote:* $P_o = [P_{o_{l,n}}]_{l\in\mathcal{L},n\in\mathcal{N}}$, *where* $P_{o_{l,n}} = \begin{cases} p_l & \text{if } l \text{ has } n \text{ for tail} \\ 0 & \text{else} \end{cases}$.

For the network described in Figure 1, we have:

$$P_o = \begin{bmatrix} 0 & 0 & 0 & p_{DA} \\ p_{AB} & 0 & 0 & 0 \\ p_{AC} & 0 & 0 & 0 \\ 0 & p_{BC} & 0 & 0 \\ 0 & p_{BD} & 0 & 0 \\ 0 & 0 & p_{CD} & 0 \end{bmatrix}$$

Property 3.2 (Equilibrium flow)**.** *For a transition matrix* $P \in \mathcal{T}$, *and demand* $D \in \mathbb{R}_+^{|\mathcal{N}|\times|\mathcal{N}|}$, $\mathbf{f}$ *is an equilibrium flow if and only if:*

a. $\mathbf{f} = P^\top \mathbf{f} + P_o(D - D^\top)\mathbf{1}$
b. $\mathbf{f}_l \geq 0 \ \forall l \in \mathcal{L}$

Proof. Let $\mathbf{f}$ be an equilibrium flow, $n \in \mathcal{N}$, and $l \in \mathcal{L}$ which has tail n. 2. from definition 3.13 gives that $f_l = p_l f_{n,out} = p_l \sum_{v,(n,v)\in\mathcal{L}} f_{n,v}$. 1. from definition 3.13 gives that $\sum_{v,(n,v)\in\mathcal{L}} f_{n,v} = \sum_{u,(u,n)\in\mathcal{L}} f_{u,n} + \sum_{\tilde{u}\in\mathcal{N}} d_{n,\tilde{u}} - \sum_{\tilde{v}\in\mathcal{N}} d_{\tilde{v},n}$.

Therefore, $f_l = p_l \left(\sum_{u,(u,n)\in\mathcal{L}} f_{u,n} + \sum_{\tilde{u}\in\mathcal{N}} d_{n,\tilde{u}} - \sum_{\tilde{v}\in\mathcal{N}} d_{\tilde{v},n}\right)$. This is equivalent to the matrix expression: $\mathbf{f} = P^\top \mathbf{f} + P_o(D - D^\top)\mathbf{1}$.

As an example, this equation will be applied to the benchmark network in Figure 1. The aim is to show that the equation results in the following system of equations:

$$\begin{aligned} f_{DA} &= p_{DA}\left(f_{BD} + f_{CD} - d_{AD}\right) \\ f_{AB} &= p_{AB}\left(f_{DA} + d_{AD}\right) \\ f_{AC} &= p_{AC}\left(f_{DA} + d_{AD}\right) \\ f_{BC} &= p_{BC} f_{AB} \\ f_{BD} &= p_{BD} f_{AB} \\ f_{CD} &= p_{CD}\left(f_{AC} + f_{BC}\right) \end{aligned}$$

Using $\mathbf{f} = P^\top \mathbf{f} + P_o(D - D^\top)\mathbf{1}$:

$$\begin{bmatrix} f_{DA} \\ f_{AB} \\ f_{AC} \\ f_{BC} \\ f_{BD} \\ f_{CD} \end{bmatrix} = \begin{bmatrix} p_{DA}(f_{BD} + f_{CD}) \\ p_{AB} f_{DA} \\ p_{AC} f_{DA} \\ p_{BC} f_{AB} \\ p_{BD} f_{AB} \\ p_{CD}(f_{AC} + f_{BC}) \end{bmatrix} + \begin{bmatrix} -p_{DA} d_{AD} \\ p_{AB} d_{AD} \\ p_{AC} d_{AD} \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} p_{DA}(f_{BD} + f_{CD} - d_{AD}) \\ p_{AB}(f_{DA} + d_{AD}) \\ p_{AC}(f_{DA} + d_{AD}) \\ p_{BC} f_{AB} \\ p_{BD} f_{AB} \\ p_{CD}(f_{AC} + f_{BC}) \end{bmatrix}$$

Remark 3.5 (Equilibrium flow)**.** *The equilibrium flow can be seen as the stationary distribution of the Markov chain described by P with the additional term $P_o(D - D^\top)\mathbf{1}$ that encodes the demand. Furthermore, the stationary distribution of a Markov chain can be seen as the limit of the Markov chain described by P as time goes to infinity (12).*

Property 3.3 (Existence of an equilibrium flow)**.** *There is at least one equilibrium flow:*

$$\forall P \in \mathscr{T},\ \forall D \in \mathbb{R}_+^{|\mathscr{N}|\times|\mathscr{N}|},\quad \mathscr{F}_D(P) \neq \emptyset$$

Proof. The proof can be done by sequentially:

1. Show that $\exists \mathbf{f}$ such that $\mathbf{f} = P^\top \mathbf{f} + P_o(D - D^\top)\mathbf{1}$.
2. Show that $\exists \mathbf{f}$ such that $\mathbf{f} = P^\top \mathbf{f} + P_o(D - D^\top)\mathbf{1}$ and $\mathbf{f} \geq 0$

1. With the notation $A = I - P^\top$ and $b = P_o(D - D^\top)\mathbf{1}$:

$$\begin{aligned} \exists \mathbf{f}, \mathbf{f} = P^\top \mathbf{f} + P_o(D - D^\top)\mathbf{1} &\iff \exists x, Ax = b \quad \text{(linear equation)} \\ &\iff b \in Im(A) \\ &\iff b \perp N(A^\top) \quad (\text{because } Im(A) \oplus^\perp N(A^\top)) \end{aligned}$$

In this case, $N(A^\top) = N(I - P) = \text{Span}(\mathbf{1})$ because of the Perron-Frobenius theorem for irreducible matrices (*13*) (basically because $P\mathbf{1} = \mathbf{1}$ and the graph is strongly connected). Therefore:

$$\begin{aligned} \exists \mathbf{f}, \mathbf{f} = P^\top \mathbf{f} + P_o(D - D^\top)\mathbf{1} &\iff \mathbf{1}^\top b = 0 \\ &\iff \mathbf{1}^\top P_o(D - D^\top)\mathbf{1} = 0 \end{aligned}$$

However, $\mathbf{1}^\top P_o = \mathbf{1}$ because of the definition of P_o and that P is a transition matrix. Hence, $\mathbf{1}^\top P_o(D - D^\top)\mathbf{1} = 0$ and $\exists \mathbf{f}, \mathbf{f} = P^\top \mathbf{f} + P_o(D - D^\top)\mathbf{1}$.

2. Let note $\mathbf{f}^*$ as a flow such that $\mathbf{f}^* = P^\top \mathbf{f}^* + P_o(D - D^\top)\mathbf{1}$. With linear algebra:

$$\{x, Ax = b\} = \{x, x = \mathbf{f}^* + y, \text{ with }, y \in N(A)\} = \mathbf{f}^* + N(A)$$

The Perron-Frobenius theorem gives the result $N(A) = N(I - P^\top) = \text{Span}(\mathbf{f}_0)$, with $\mathbf{f}_0 > 0$.
Therefore, $\{\mathbf{f}, \mathbf{f} = P^\top \mathbf{f} + P_o(D - D^\top)\mathbf{1}\} = \{\mathbf{f}, \mathbf{f} = \mathbf{f}^* + \alpha \mathbf{f}_0, \alpha \in \mathbb{R}\}$.
Because $\mathbf{f}_0 \geq = 0$, then $\mathscr{F}_D(P) = \{\mathbf{f}, \mathbf{f} = \mathbf{f}^* + \alpha \mathbf{f}_0, \alpha \in \mathbb{R}\} \cap \{\mathbf{f}, \mathbf{f} \geq 0\} \neq \emptyset$.

Now, $\mathbf{1}^\top P_o(D - D^\top)\mathbf{1} = 0$ can be shown on the benchmark example in Figure 1.

$$
\begin{aligned}
\mathbf{1}^\top P_o (D - D^\top)\mathbf{1} &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -p_{DA}d_{AD} \\ p_{AB}d_{AD} \\ p_{AC}d_{AD} \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
&= -p_{DA}d_{AD} + p_{AB}d_{AD} + p_{AC}d_{AD} \\
&= -d_{AD} + d_{AD} && \text{since } (p_{AB} + p_{AC}) = 1 \\
&= 0
\end{aligned}
$$

Remark 3.6 (Uniqueness of an equilibrium flow, convergence of Markov chain)**.** *The equilibrium flow is not unique. However, one can fix some conditions on link flows such that the equilibrium flow will be unique and is the limit of the Markov chain described by P (c.f. remark 3.5).*

On the benchmark network example (Figure 1), one possible constraint is $f_{DA} = 0$. In this case, the equilibrium will be uniquely defined (and will still satisfy $\mathbf{f} \geq 0$).

Remark 3.7 (Feasibility of an equilibrium flow)**.** *An equilibrium flow (definition 3.2) may not be a feasible flow (definition 3.6). This is due to the potential presence of cycles in the network, which was only accounted for as a constraint such that paths should not have cycles (see Figure 3).*

However, on the benchmark example, if one imposes the constraint that $f_{DA} = 0$, then the equilibrium flow will be feasible.

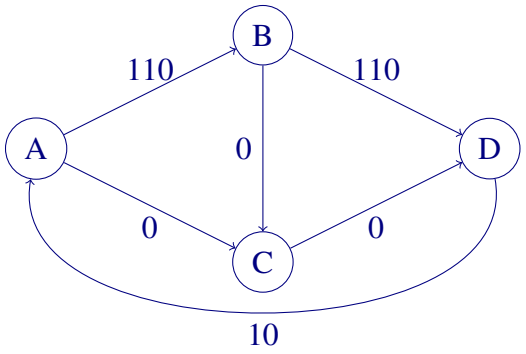


FIGURE 3: An equilibrium flow which is not feasible. Given a demand of $d_{AD} = 100$, the flow described in the figure is an equilibrium flow but is not a feasible flow. For any feasible flow allocation $f_{DA} = 0$.

OPTIMAL STATIC FLOW ALLOCATION

In this section, the user equilibrium and social optimum flow allocation are defined for the classic static traffic assignment (*10*) and for the Markovian framework. In the case of user equilibrium, vehicles are assumed to try to reduce their travel times as much as possible (*15*). Intuitively, this means that at user equilibrium, no user can switch to a different path and reduce his or her travel

time (*15*). In the case of the social optimum allocation, the total of all vehicles' travel times in the network is minimized (given a fixed demand). Both the user equilibrium and the social optimum, for the classic static traffic assignment framework and the Markovian framework, can be computed as minimization problems.

Static traffic assignment
First, recall the classic static traffic assignment user equilibrium and social optimum (*10*).

Definition 4.1 (User equilibrium)**.** *Given a traffic demand D, a path flow allocation* $\mathbf{h} \in \mathscr{H}_D$ *is a user equilibrium if and only if:*

$$\forall o,d \in \mathscr{N},\ \forall \mathbf{p} \in \mathscr{P}_{od},\ h_{\mathbf{p}} \cdot (c_{\mathbf{p}}(\mathbf{h}) - \min_{\mathbf{q} \in \mathscr{P}_{od}} c_{\mathbf{q}}(\mathbf{h})) = 0$$

Definition 4.2 (Social optimum)**.** *Given a traffic demand D, a link flow allocation* $\mathbf{f} \in \mathscr{F}_D$ *is a social optimum if and only if:*

$$\forall \mathbf{g} \in \mathscr{F}_D,\ \mathbf{t}(\mathbf{f})^\top \mathbf{f} \leq \mathbf{t}(\mathbf{g})^\top \mathbf{g}$$

Remark 4.1 (Wardrop's first condition (*16*))**.** *At a* ***user equilibrium****, the travel time on all used paths between an origin-destination pair are equal and less than those which would be experienced by a single vehicle on any unused path in the network.*

Property 4.1 (Minimization problem to compute user equilibrium)**.** *Any user equilibrium is the solution of the following a convex optimization problem (17, 18):*

$$\begin{aligned}
&\min_{\mathbf{f},\mathbf{h}} \sum_{l \in \mathscr{L}} \int_0^{f_l} t_l(s)ds && \\
&\textit{subject to} \sum_{\mathbf{p} \in \mathscr{P}} \delta_{l,\mathbf{p}} h_{\mathbf{p}} = f_l && \forall l \in \mathscr{L} \\
&\qquad \sum_{\mathbf{p} \in \mathscr{P}_{od}} h_{\mathbf{p}} = d_{od} && \forall o,d \in \mathscr{N} \\
&\qquad h_{\mathbf{p}} \geq 0 && \forall \mathbf{p} \in \mathscr{P}
\end{aligned}$$

This can be written as:

$$\max_{\mathbf{f} \in X} R(\mathbf{f})$$

Where $X = \mathscr{F}_D$ *and* $R(\mathbf{f}) = -\sum_{l \in \mathscr{L}} \int_0^{f_l} t_l(s)ds$.

Property 4.2 (Minimization problem to compute social optimum)**.** *Any user equilibrium is the solution of the following a convex optimization problem (10):*

$$\begin{aligned} &\min_{\mathbf{f},\mathbf{h}} \mathbf{t}(\mathbf{f})^\top \mathbf{f} \\ \textit{subject to } &\sum_{\mathbf{p}\in\mathscr{P}} \delta_{l,\mathbf{p}} h_{\mathbf{p}} = f_l && \forall l \in \mathscr{L} \\ &\sum_{\mathbf{p}\in\mathscr{P}_{od}} h_{\mathbf{p}} = d_{od} && \forall o,d \in \mathscr{N} \\ &h_{\mathbf{p}} \geq 0 && \forall \mathbf{p} \in \mathscr{P} \end{aligned}$$

This can be written as:

$$\max_{\mathbf{f}\in X} R(\mathbf{f})$$

Where $X = \mathscr{F}_D$ *and* $R(\mathbf{f}) = -\mathbf{t}(\mathbf{f})^\top \mathbf{f}$.

Optimal flow in the Markovian framework

The user equilibrium and social optimum notions can be extended to the Markovian framework. The idea is to perform the same minimization problems but on the set of equilibrium flows given by the transition matrix, instead of the set of feasible flow allocations. The motivation behind the new Markovian formulation is to reduce the number of variables in the static traffic assignment. Instead of computing every path flow (which may be expensive), the Markovian framework only requires information on the split ratio at each intersection. Therefore, an advantage of the Markovian framework is the ability to perform decentralized traffic control on routing choices.

Definition 4.3 (User equilibrium in Markovian framework)**.** *Given a traffic demand D, a flow allocation* $\mathbf{f}$ *is a user equilibrium in the Markov framework if it is solution of:*

$$\begin{aligned} &\min_{\mathbf{f},P} \sum_{l\in\mathscr{L}} \int_0^{f_l} t_l(s)ds \\ \textit{subject to } &\mathbf{f} = P^\top \mathbf{f} + (P_O D - P_O D^\top)\mathbf{1} \\ &\mathbf{f} \geq 0 \\ &P \in \mathscr{T} \end{aligned}$$

This can be written as:

$$\max_{\mathbf{f}\in X} R(\mathbf{f})$$

Where $X = \bigcup_{P\in\mathscr{T}} \mathscr{F}_D(P)$ *and* $R(\mathbf{f}) = -\sum_{l\in\mathscr{L}} \int_0^{f_l} t_l(s)ds$.

Definition 4.4 (Social optimum in Markovian framework)**.** *Given a traffic demand D, a link flow*

allocation $\mathbf{f}$ *is a social optimum in the Markov framework if it is a solution of:*

$$\begin{aligned} & \min_{\mathbf{f},P} \mathbf{t}(\mathbf{f})^\top \mathbf{f} \\ \textit{subject to} \quad & \mathbf{f} = P^\top \mathbf{f} + (P_O D - P_O D^\top)\mathbf{1} \\ & \mathbf{f} \geq 0 \\ & P \in \mathscr{T} \end{aligned}$$

This can be written as:

$$\max_{\mathbf{f} \in X} R(\mathbf{f})$$

Where $X = \bigcup_{P \in \mathscr{T}} \mathscr{F}_D(P)$ *and* $R(\mathbf{f}) = -\mathbf{t}(\mathbf{f})^\top \mathbf{f}$.

Remark 4.2 (Equivalence between both frameworks)**.** *The reader may expect the optimal flow allocation of the Markovian framework to be the same as the optimal flow allocation of the classic STA framework. However, remark 3.7 shows that the set of feasible flow allocations and the set of equilibrium flows over every transition matrix may be different. Nevertheless, if the optimal flow allocation in the Markovian framework is a feasible flow allocation and if the optimal flow allocation in the STA framework is an equilibrium flow allocation, then definitions 4.1 and 4.3 and definitions 4.2 and 4.4 will be equivalent.*

DYNAMIC TRAFFIC ASSIGNMENT

In contrast to the static traffic assignment, dynamic models can simulate time-varying networks and establish new equilibria for different departure times. Therefore, the previously defined Markovian framework is extended to the dynamic traffic assignment using SUMO. Optimization frameworks are developed based on the dynamic notion of the user equilibrium (minimizing average marginal regret) and the social optimum (minimizing the total vehicles hours of travel). The benchmark example is then set up in SUMO.

Limitations of static traffic assignment framework

Static models are defined over a relatively long period of time, and the congestion properties of each link are described by a volume-delay function (VDF) that expresses the average link travel time as a function of the traffic volume on that link. The travel time on each path would be the summation of the travel time on each link in the path (as described in definition 3.7). Moreover, the static model cannot model node dynamics, and for each link, the inflow is always equal to the outflow in the static model (as described in definition 3.12). Such features cannot capture the dynamics of congestion in each link: how congestion is propagated upstream through the link and spills back to the upstream link (*19*). Furthermore, the static traffic assignment assumes that the travel demand for each origin-destination pair is uniformly distributed over time (as defined in definition 3.1) (*20*). Consequently, the static traffic assignment cannot model scenarios with changing demand.

Optimization framework for dynamic traffic assignment

Advantages of dynamic traffic assignment

Recognizing that traffic networks are generally not in the steady state, extending the notion of user equilibrium from the static traffic assignment to dynamic traffic assignment (*19*) requires:

1. Developing efficient ways of describing time-varying network traffic conditions and finding the path with the shortest travel time, considering that link travel times changes over time.
2. Establishing a new equilibrium for each departure time.

Compared to static traffic assignment models, dynamic traffic assignment models are able to capture more realistic traffic flow characteristics such as queue spillback, expansion waves, and changing demand (*21*). In addition, the traditional travel forecasting models perform simulations on static regional traffic flow while microscopic traffic simulation models focus on dynamic corridor-level traffic analysis (*19*). The dynamic traffic assignment models' ability to model at a wide range of scales from the corridor-level to the regional-level fills the gap between travel forecasting models and microscopic traffic simulation models (*19*). This can significantly improve the accuracy of traffic estimation.

In this article, a simulation-based dynamic traffic assignment model, using a traffic simulator, reproduces the flow dynamics in urban road traffic systems and collects simulation data for optimization (*22*). A microscopic and open-source simulator SUMO (*23*) simulates the traffic flow propagation and spatial-temporal interaction. SUMO is able to handle easy query and traffic control with Python API traci.

User equilibrium and social optimum extensions in the dynamic traffic assignment

The Markovian framework can be extended using SUMO. First, given a transition matrix, the set of equilibrium flows $\mathscr{F}_D(P)$ can now be computed.

Definition 5.1 (Flow allocation using SUMO)**.** *Using SUMO, given a transition matrix $P \in \mathscr{T}$ and demand $D \in \mathbb{R}_+^{|\mathscr{N}|\times|\mathscr{N}|}$, we denote $\tilde{\mathscr{F}}_D(P)$ as the set of corresponding steady-state flow allocations that SUMO computes with P and D given to the simulator.*

Remark 5.1 (Steady-state flow allocation)**.** *Due to the stochastic nature of the simulation process, it generally takes some steps for the flow allocation to achieve a steady state given a fixed transition matrix which encodes the nodes' split ratios. As demonstrated in Figure 5, as the simulation evolves, it takes a certain amount of time steps for the reward distributions (and therefore the flow allocations) to stabilize.*

Then, using $\tilde{\mathscr{F}}_D(P)$, definitions 4.3 and 4.4 can be extended to the dynamic case.

Definition 5.2 (Social optimum in Markovian dynamic framework)**.** *Given a traffic demand D, a transition matrix P defines a social optimum in the Markov dynamic framework if it is solution of:*

$$\begin{aligned} &\max_{\mathbf{f},P} R(\mathbf{f}) \\ \textit{subject to} \quad &\mathbf{f} \in \tilde{\mathscr{F}}_D(P) \\ &R(\mathbf{f}) = -\mathbf{t}(\mathbf{f})^\top \mathbf{f} \\ &P \in \mathscr{T} \end{aligned}$$

Where $\mathbf{t}(\mathbf{f})^\top \mathbf{f}$ describes the total vehicle hour travelled in the network.

Definition 5.3 (User equilibrium in Markovian dynamic framework)**.** *Given a traffic demand D, a transition matrix P defines a user equilibrium in the Markov dynamic framework if it is a solution*

of:

$$\begin{aligned} &\max_{\mathbf{f},P} R(\mathbf{f}) \\ \textit{subject to} \quad &\mathbf{f} \in \tilde{\mathscr{F}}_D(P) \\ &R(\mathbf{f}) = -\max_{\mathbf{x}\in\mathscr{F}_D} \mathbf{t}(\mathbf{f})^\top(\mathbf{f}-\mathbf{x}) \\ &P \in \mathscr{T} \end{aligned}$$

$R(\mathbf{f}) = \max_{\mathbf{x}\in\mathscr{F}_D} \mathbf{t}(\mathbf{f})^\top(\mathbf{f}-\mathbf{x})$ *expresses the average marginal regret of vehicles in the network (24). It has been shown that minimizing* $\max_{\mathbf{x}\in\mathscr{F}_D} \mathbf{t}(\mathbf{f})^\top(\mathbf{f}-\mathbf{x})$ *and minimizing* $\sum_{l\in\mathscr{L}} \int_0^{f_l} t_l(s)ds$ *are equivalent (18, 24). Furthermore, the average marginal regret can be extended to the dynamic traffic state (25). Intuitively, the average marginal regret measures the distance from a traffic flow allocation to a Nash equilibrium, or user equilibrium (24).*

Reward functions

Based on the two previous definitions, we experimented with three reward functions:

1. **Average throughput**: maximizing f_l at exit:

$$R_{thru} = f_{Exit}, \tag{1}$$

   where f_{Exit} is defined in section "SUMO Setup".

2. **Social optimum**: maximizing the inverse of vehicle hours of travel (VHT), that is, the sum of link flows weighted by average travel time of the edge divided by the sum of link flows:

$$R_{VHT} = \frac{1}{VHT}, \quad VHT = \frac{\sum_{l\in\mathscr{L}}(f_l+1)\cdot t_l}{\sum_{l\in\mathscr{L}} f_l + 1}, \quad t_l = \frac{s_l}{v_l + 0.1v_{l\max}}, \tag{2}$$

   where v_l is average speed on link l and $v_{l\max}$ is the speed limit on link l. Note that addition of one to flow and $0.1v_{l\max}$ to speed is to prevent from division by zero and is not present in the original theoretical definition of those terms.

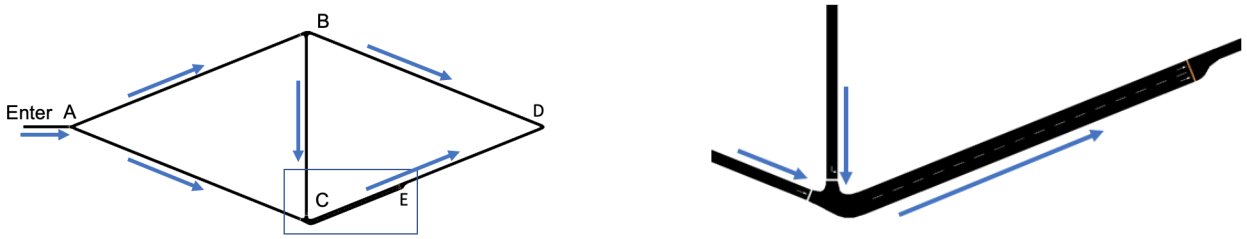
3. **User equilibrium**: maximizing inverse of average marginal regret (*24*) divided by the total demand in the network:

$$R_{regret} = \frac{\sum_{\mathbf{p}\in\mathscr{P}}(h_\mathbf{p}+1)(c_\mathbf{p} - \min_{\tilde{\mathbf{p}}} c_{\tilde{\mathbf{p}}})}{\sum_{\mathbf{p}\in\mathscr{P}} h_\mathbf{p} + 1}, \quad c_\mathbf{p} = \frac{s_\mathbf{p}}{v_\mathbf{p} + 0.1v_{\mathbf{p}\max}}, \tag{3}$$

   where $v_\mathbf{p}$ is the average speed on path $\mathbf{p}$ and $v_{\mathbf{p}\max}$ the speed limit on link p. Similar to R_{VHT}, we add one to h_p and $0.1v_{\mathbf{p}\max}$ to speed for numerical stability. Furthermore, in our case study, we approximate traffic on path *ABD* with traffic on path *BD*; traffic on path *ABCED* with traffic on path *BC*, and traffic on path *ACED* with traffic on path *AC*.

SUMO setup

The road network in Figure 1 is simulated in SUMO (*26*). The goal here is to simulate the traffic states with different split ratios and to provide data for flow allocation optimization.



(a) SUMO Network Setup: A screenshot from SUMO GUI that describes the geometry of nodes and edges. Blue arrows are added to visualize the direction of traffic flow.

(b) Merge dynamics: A zoom out window that details how the merge at C is designed on the right network. Note that vehicles from the two lanes take turns to merge in a "first come, first served" basis.

FIGURE 4: SUMO Setup

The simulated traffic network is illustrated in Figure 4 and the simulation setup is detailed. For source code, please refer to (*27*).

1. Set time step $\Delta t = 0.5s$.
2. Generate a demand profile d using a Poisson distribution with a rate of 1 veh/s.
3. Input a transition matrix $P \in \mathscr{T}$.
4. Initialize the network with no vehicles.
5. For every time step $i \in T$:
   - Generate vehicles $v \in \mathscr{V}_i$ for the time step i given the demand (where $\mathscr{V}_i$ is the set of all vehicles generated at the time step i, $|\mathscr{V}_i| = d$).
   - For every vehicle generated:
     - Create vehicle path according to transition matrix.
     - Add vehicle v to the network following path $\mathbf{p}_v$.

LEARNING OPTIMAL FLOW ALLOCATION

In this section, we describe the three optimization methods, i.e., grid search, random search, and evolution strategies, we used to search for optimal transition matrix under the three aforementioned reward functions, i.e., average throughput, social optimum, and user equilibrium. As illustrated in Figure 4, the road network has two splits at n_A and n_B. Therefore, we only need to optimize two variables, p_{AB} and p_{BC}.

As a baseline, we start with a very simple learning algorithm grid search. Next, we solve the same optimization problem using random search and then, a more advanced optimizer, evolution strategies, respectively. Finally, we remark on their pros and cons and discuss how to choose among them in practice.

Grid search

Grid search is an optimization procedure that finds an optimal solution by checking the rewards at every point in a discretized search space and selecting the point(s) with the highest reward. Pseudocode of this algorithm is described in Algorithm 1(*28*)

The results of grid search is illustrated in Figure 5. To illustrate evolution of the system dynamics, we plot the heatmap of each reward on a meshgrid of p_{AB} and p_{BC} with each axis

incrementing from 0 to 1 at stepsize of 0.05. Each simulation is designed to execute 1000 steps, i.e., 500 seconds; each column of Figure 5 shows the progression of heatmaps for each reward function throughout the simulation timestamped at 37.5 s, 87.5 s, 287.5 s, and 487.5 s.

Reward functions are evaluated in equilibrium states using data between 475 s and 500 s. The resulting optima found by grid search are marked in the last row of the figure with "$\star$" symbols.

The interpretation of plot of average throughput reward is straightforward because in order to maximize the the exit flow, it's reasonable to minimize congestion in every path. As illustrated in Figure 4, there is a merge at C. When a vehicle is about to turn at a sharp angle, there is a speed reduction for all vehicles in the area, which increases the path travel time. When the amount of vehicles trying to merge at C exceeds capacity, the speed reduction would result in spillbacks in AC or/and BC and therefore decrease the flow rate at exit. In consideration of the merge, when $p_{AB} >= 0.6$ and $p_{BC} <= 0.4$, congestion is minimized and the exit flow maximized.

The plot for social optimum reward is also reasonable because in order to minimize the weighted average flow travel time, we want to keep each $\mathbf{p} \in \mathscr{P}$ that is in use during simulation unsaturated so the average travel time would be minimized. Likewise, we should take the speed reduction at sharp turns into account. The turning angle from AB to BC and from BC to CE is larger than that from AC to CE, therefore vehicles would be more likely to experience a larger the speed reduction if taking the route of AB than that of AC. When p_{AB} is zero, which means all cars are taking the $\mathbf{p}_{Enter \to A \to C \to E \to D}$, the reward would be maximized. The worst scenario would be about half percentage of incoming vehicles takes AB and most of these vehicles turn into BC at B to merge with the other half of the vehicles taking AC at C. Such merge would cause serious spill back that significantly elongate the path travel time, and therefore the weighted average flow travel time.

The plot for user equilibrium reward is compatible with the characteristics of the network. The user equilibrium achieves when no one in the network could achieve a shorter travel time if switching to a different path. Therefore, when vehicles approximately have equal split at both A and B, there would be queues spilling ack upstream to both BC and AB. Such congestion delay would balance out the difference in path's free flow travel time and creates the social equilibrium that each user experiences the same travel time. If some paths in the network is congested while one path is unsaturated, some vehicles could always achieve shorter travel time by switching to the other path. If one path is under-saturated, some vehicles could always achieve shorter travel time by switching to the other path. Note that the maximum point is slightly biased toward AB and BD. Such bias is also an artefact of reduced road capacity at BC and CD induced by sharp turning angles.

Random search

Random search is an optimization method that finds an optimal solution by checking the rewards at a few random points from search space and selecting the point(s) with the highest reward. Pseudocode of this algorithm is described in Algorithm 2 (*29*).

The result is illustrated in Figure 6a. In the figure, 50 random points are sampled, among which the best point is marked by "+" symbol. The location marked by "$\star$" symbol is the optimal solution found by grid search. By visually checking the graph, the solutions found by the two methods are comparable.

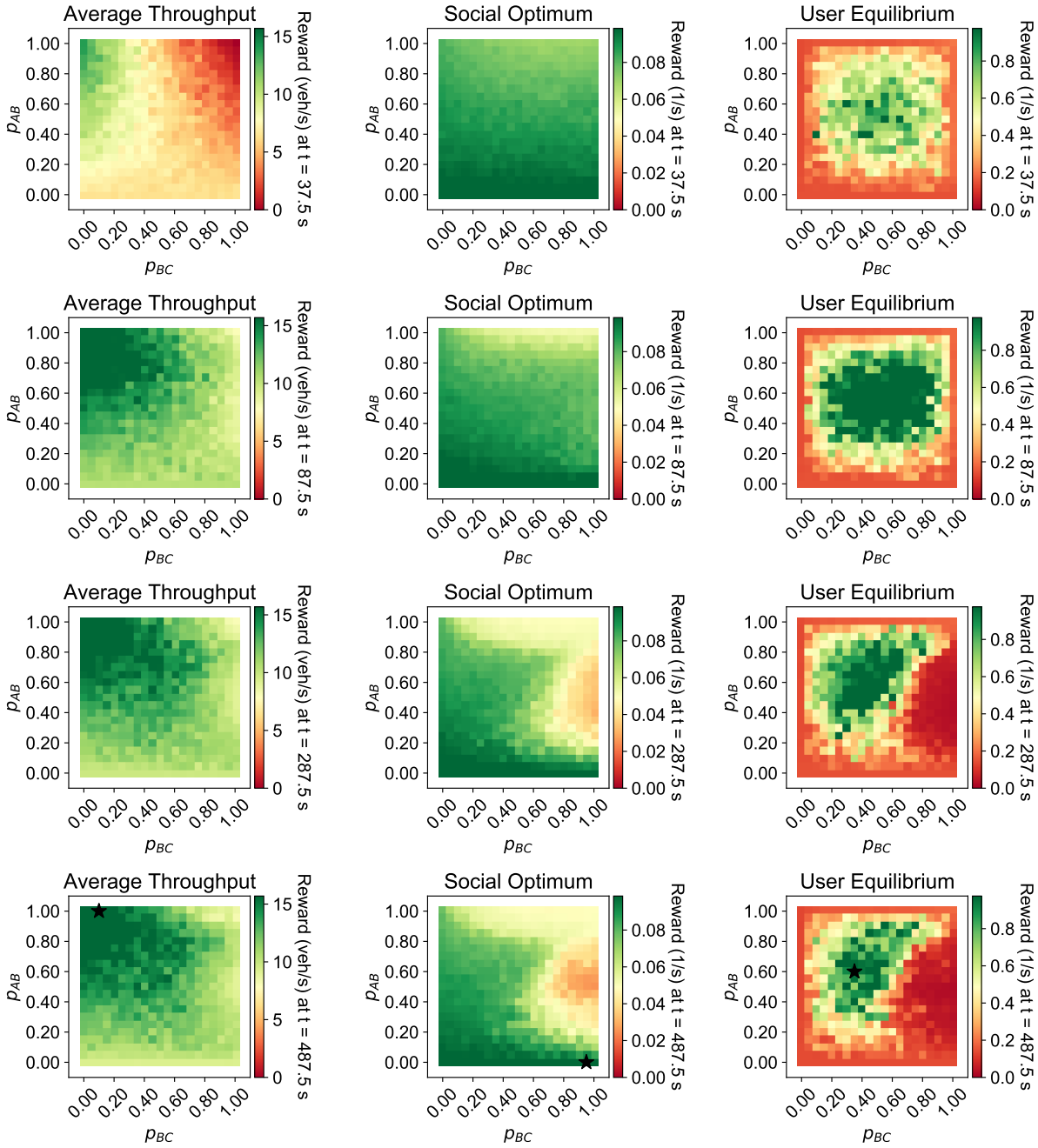
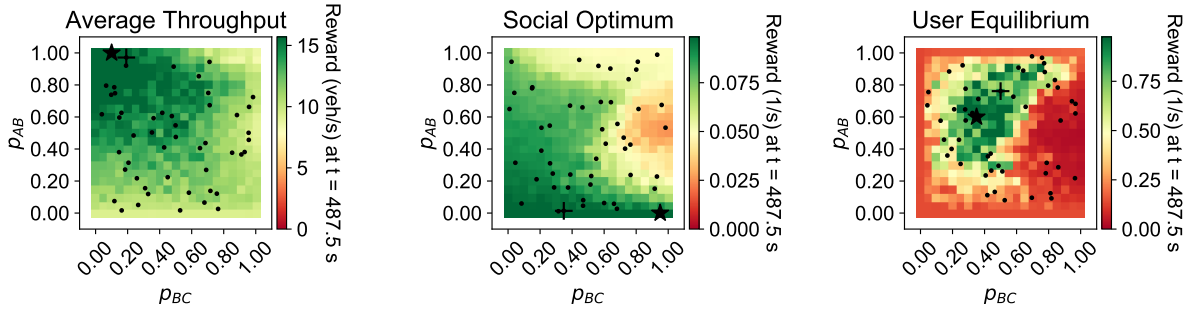
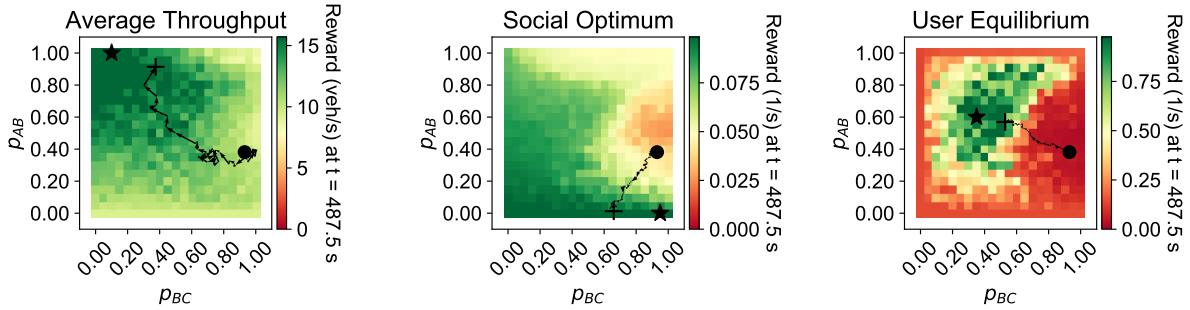


FIGURE 5: Grid Search Results: Each column corresponds to the time-varying distribution of each reward function on the meshgrid of P_{AB} and P_{BC} throughout the simulation. The three columns respectively correspond to reward functions for average outflow, inverse of vehicle hours of travel and inverse of average marginal regret. At time 487.5s, when the network has converged to steady state, the point that gives the highest reward is marked as $\star$ for each reward function.



(a) Random Search: Random search graph is performed on top of the reward distribution graph at 975 time-steps of simulation. 50 sample points (black dots) are picked to measure the reward function and the optimal point that gives the highest reward is marked by "+" symbol. For comparison, the solution found by grid search is marked by "⋆" symbol.



(b) Evolution Strategies: Similar to random search, based on the reward distribution at 975 time-steps of simulation, evolution strategies first randomly pick a starting point and then start climbing uphill with gradient ascent demonstrated by the black trace. The algorithm is manually terminated after 50 iterations at location marked by "+" symbol. For comparison, the solution found by grid search is marked by "⋆" symbol.

FIGURE 6: Random Search and Evolution Strategies Results

Evolution strategies

Evolution strategies is an optimization method that finds an optimal solution using gradient ascent, where the gradients are estimated from Gaussian sampling in the neighbourhood of current solution. Specifically, for each iteration, a stochastic gradient is computed from rewards of a few points normally distributed around the current solution. The gradient is then added to the current solution to generate the new solution. Pseudocode of this algorithm is described in Algorithm 3 (*30*).

The results of evolution strategies for three different reward functions is illustrated in Figure 6b. For a fair comparison with the random search method, the trajectories are stopped after 50 iterations. All three trajectories, despite being occasionally distracted, are generally climbing uphill toward higher rewards. The termination locations are marked by "+" symbol. Comparing them with those marked by "$\star$" symbol, the optimal points found by evolution strategies are just as good as the ones found by grid search.

Algorithm 1: Grid Search

Policy $\theta \leftarrow (p_{AB}, p_{BC})$;
Reward $R(\theta)$;
Search step size Δ;
Samples per point n;
Time horizon T;
Warmup time τ;
$\hat{r^*} \leftarrow -\infty$;
for $p_{AB} \leftarrow 0$; $p_{AB} \leq 1$; $p_{AB} \leftarrow p_{AB} + \Delta$ **do**
  for $p_{BC} \leftarrow 0$; $p_{BC} \leq 1$; $i \leftarrow p_{BC} + \Delta$ **do**
    $\bar{\theta} \leftarrow (p_{AB}, p_{BC})$;
    for $i \leftarrow 0$; $i < n$; $i \leftarrow i+1$ **do**
      $r_i \leftarrow 0$;
      for $t \leftarrow 0$; $t < T$; $t \leftarrow t+1$ **do**
        Step simulation;
        if $t > \tau$ **then**
          $r_i \leftarrow r_i + R(\bar{\theta})$;
        end
      end
    end
    $\bar{r} \leftarrow \frac{1}{n}\sum_{i\leftarrow 0}^{n-1} r_i$
    if $\bar{r} > \hat{r^*}$ **then**
      $\hat{r^*} \leftarrow \bar{r}$;
      $\hat{\theta}^* \leftarrow \bar{\theta}$;
    end
  end
end
return $\hat{\theta}^*$

Algorithm 2: Random Search

Policy $\theta \leftarrow (p_{AB}, p_{BC})$;
Reward $R(\theta)$;
Number of random samples m;
Samples per point n;
Time horizon T;
Warmup time τ;
$\hat{r^*} \leftarrow -\infty$;
for $j \leftarrow 0$; $j < m$; $j \leftarrow j+1$ **do**
  $\bar{\theta} \leftarrow (p_{AB}, p_{BC})$ where $p_{AB}, p_{BC} \sim \mathscr{U}(0,1)$;
  for $i \leftarrow 0$; $i < n$; $i \leftarrow i+1$ **do**
    $r_i \leftarrow 0$;
    for $t \leftarrow 0$; $t < T$; $t \leftarrow t+1$ **do**
      Step simulation;
      if $t > \tau$ **then**
        $r_i \leftarrow r_i + R(\bar{\theta})$;
      end
    end
  end
  $\bar{r} \leftarrow \frac{1}{n}\sum_{i\leftarrow 0}^{n-1} r_i$
  if $\bar{r} > \hat{r^*}$ **then**
    $\hat{r^*} \leftarrow \bar{r}$;
    $\hat{\theta}^* \leftarrow \bar{\theta}$;
  end
end
return $\hat{\theta}^*$

Algorithm 3: Evolution Strategies

Policy $\theta \leftarrow (p_{AB}, p_{BC})$;
Reward $R(\theta)$;
Number of iterations m;
Learning rate α;
Noise magnitude σ;
Samples per iteration n;
Time horizon T;
Warmup time τ;
Initialize $\hat{\theta}^* \leftarrow (p_{AB}, p_{BC})$ where $p_{AB}, p_{BC} \sim \mathscr{U}(0,1)$;
for $j \leftarrow 0$; $j < m$; $j \leftarrow j+1$ **do**
  for $i \leftarrow 0$; $i < n$; $i \leftarrow i+1$ **do**
    Sample $\varepsilon_i \sim \mathscr{N}(0, \sigma I)$;
    $\theta_i \leftarrow \hat{\theta}^* + \varepsilon_i$;
    Clip all θ_i entries to be within [0, 1];
    $r_i \leftarrow 0$;
    for $t \leftarrow 0$; $t < T$; $t \leftarrow t+1$ **do**
      Step simulation;
      if $t > \tau$ **then**
        $r_i \leftarrow r_i + R(\bar{\theta})$;
      end
    end
  end
  $\nabla\hat{\theta}^* \leftarrow \frac{1}{n\sigma}\sum_{i\leftarrow 0}^{n-1} r_i \varepsilon_i$;
  $\hat{\theta}^* \leftarrow \hat{\theta}^* + \alpha\nabla\hat{\theta}^*$;
end
return $\hat{\theta}^*$

Discussion

Although grid search can be exhaustive, it does not scale well with the dimension of the problem. This is commonly known as the "curse of dimensionality." Compared to grid search, random search is found to be more efficient, both theoretically and empirically (*31*). Compared to random search, evolution strategies prefers samples nearby and may perform better in problems where local reward topology is constructive to guide gradient ascent.

In practice, if the dimension of the problem is small, all of the three methods are good choices. For large problems, random search or evolution strategies should be used instead. For problems where it is possible to efficiently compute the exact gradient, vanilla gradient ascent methods may be used instead. However, random search and evolution strategies can be easily parallelized across a large cluster of nodes, which may result in finding a good solution within less wall clock time.

CONCLUSION

This article first compares the Markovian static traffic assignment to the classic static traffic assignment. The notions of user equilibrium and social optimum flow allocations are introduced for the Markovian framework. The Markovian framework reduces the number of variables to compute in the static traffic assignment by introducing the Markov property of the traffic system (i.e. a traveler does not take into account her/his previous path to decide the next road to take). Then, extensions of these notions to the dynamic traffic assignment using the traffic micro-simulator SUMO are made. On a benchmark network, the simulator learns optimal routing behavior by finding optimal flow split ratios at every node using grid search, random search, and evolution strategies. The results show the ability of the Markovian framework to learn optimal routing behavior using SUMO.

The work can lead to several potential applications. First, the transition matrix framework may be used to perform traffic estimation through data assimilation using regression with cross-sectional data. Second, decentralized traffic control strategies may be developed by learning optimal transition matrices. Third, Markov chain theory and graph theory can be used to better understand the resiliency of the traffic network. Lastly, time-varying transition matrices in which the optimal split ratios may be learned through reinforcement learning will be considered. Another direction will be to consider split ratios that depend on the destinations of the vehicles, where every possible destination has a transition matrix associated with it.

REFERENCES

[1] Teodorovic, D. and M. Janic, Chapter 8 - Transportation Demand Analysis. In *Transportation Engineering* (D. Teodorovic and M. Janic, eds.), Butterworth-Heinemann, 2017, pp. 495 – 568.

[2] *INRIX Global Traffic Scorecard*, INRIX Research, 2018.

[3] Barth, M. and K. Boriboonsomsin, Traffic congestion and greenhouse gases, 2009.

[4] Cabannes, T., M. A. Sangiovanni Vincentelli, A. Sundt, H. Signargout, E. Porter, V. Fighiera, J. Ugirumurera, and A. M. Bayen, The impact of GPS-enabled shortest path routing on mobility: a game theoretic approach. Transportation Research Board 97th Annual Meeting, 2018.

[5] Patire, A. D., M. Wright, B. Prodhomme, and A. M. Bayen, How much GPS data do we need? *Transportation Research Part C: Emerging Technologies*, Vol. 58, 2015, pp. 325–342.

[6] Crisostomi, E., S. Kirkland, and R. Shorten, A Google-like model of road network dynamics

and its application to regulation and control. *International Journal of Control*, Vol. 84, No. 3, 2011, pp. 633–651.

[7] Wu, C., A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, Flow: Architecture and benchmarking for reinforcement learning in traffic control. *arXiv preprint arXiv:1710.05465*, 2017.

[8] Langville, A. N. and C. D. Meyer, *Google's PageRank and beyond: The science of search engine rankings*. Princeton University Press, 2011.

[9] Zhang, Z. and Q. Chen, Comparison of the Eulerian and Lagrangian methods for predicting particle transport in enclosed spaces. *Atmospheric environment*, Vol. 41, No. 25, 2007, pp. 5236–5248.

[10] Patriksson, M., *The traffic assignment problem: models and methods*. Courier Dover Publications, 2015.

[11] Zhao, X. and J. C. Spall, A Markovian framework for modeling dynamic network traffic. In *2018 Annual American Control Conference (ACC)*, IEEE, 2018, pp. 6616–6621.

[12] Gagniuc, P. A., *Markov Chains*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2017.

[13] Godsil, C. and G. Royle, *Algebraic Graph Theory*, Vol. 207 of *Graduate Texts in Mathematics*. Springer New York, New York, NY, 2001.

[14] Porta, S., P. Crucitti, and V. Latora, The network analysis of urban streets: a dual approach. *Physica A: Statistical Mechanics and its Applications*, Vol. 369, No. 2, 2006, pp. 853–866.

[15] Perederieieva, O., M. Ehrgott, J. Y. Wang, and A. Raith, A computational study of traffic assignment algorithms. In *Australasian Transport Research Forum, Brisbane, Australia*, 2013, pp. 1–18.

[16] Wardrop, J. G., Some Theoretical Aspects of Road Traffic Research. In *ICE Proceedings: Engineering Divisions*, 1952, Vol. 1, pp. 325–362.

[17] Beckmann, M., C. B. McGuire, and C. B. Winsten, *Studies in the Economics of Transportation*. Yale University Press, 1956.

[18] Nagurney, A., *Network Economics: A Variational Inequality Approach*. Advances in Computational Economics, Springer US, 1998.

[19] Chiu, Y.-C., J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, and J. Hicks, *Dynamic Traffic Assignment A primer*. Transporation Research Circular Number E-C153, 2011.

[20] Saw, K., B. Katti, and G. Joshi, Literature review of traffic assignment: static and dynamic. *International Journal of Transportation Engineering*, Vol. 2, No. 4, 2015, pp. 339–347.

[21] Szeto, W. and H. K. Lo, Dynamic Traffic Assignment: Properties and Extensions. *Transportmetrica*, Vol. 2, No. 1, 2006, pp. 31–52.

[22] Peeta, S. and A. K. Ziliaskopoulos, Foundations of Dynamic Traffic Assignment: The Past, the Present and the Future. *Networks and Spatial Economics*, Vol. 1, No. 3, 2001, pp. 233–265.

[23] Krajzewicz, D., G. Hertkorn, C. Rössel, and P. Wagner, SUMO (Simulation of Urban MObility)-an open-source traffic simulation. In *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, 2002, pp. 183–187.

[24] Cabannes, T., M. Sangiovanni, A. Keimer, and A. M. Bayen, Regrets in Routing Networks: Measuring the Impact of Routing Apps in Traffic. *ACM Trans. Spatial Algorithms Syst.*, Vol. 5, No. 2, 2019, pp. 9:1–9:19.

[25] Cabannes, T., F. Shyu, E. Porter, S. Yao, Y. Wang, M. A. S. Vincentelli, S. Hinardi, M. Zhao, and A. M. Bayen, Measuring regret in routing: assessing the impact of increased app usage.

In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 2589–2594.

[26] Krajzewicz, D., J. Erdmann, M. Behrisch, and L. Bieker, Recent Development and Applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, Vol. 5, No. 3&4, 2012, pp. 128–138.

[27] Fangyu Wu, J. L., *Learning Matrix*. `https://github.com/lijiayi9712/Learning_matrix.git`, 2019.

[28] Jiménez, Á. B., J. L. Lázaro, and J. R. Dorronsoro, *Finding Optimal Model Parameters by Discrete Grid Search*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 120–127, 2007.

[29] Zabinsky, Z. B., *Random Search Algorithms*, American Cancer Society, 2011.

[30] Salimans, T., J. Ho, X. Chen, S. Sidor, and I. Sutskever, Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[31] Bergstra, J. and Y. Bengio, Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, Vol. 13, No. Feb, 2012, pp. 281–305.