A Bio-Inspired Approach to Traffic Network Equilibrium Assignment Problem

Xiaoge Zhang and Sankaran Mahadevan

***Abstract*—Finding an equilibrium state of the traffic assignment plays a significant role in the design of transportation networks. We adapt the path finding mathematical model of slime mold *Physarum polycephalum* to solve the traffic equilibrium assignment problem. We make three contributions in this paper. First, we propose a generalized *Physarum* model to solve the shortest path problem in directed and asymmetric graphs. Second, we extend it further to resolve the network design problem with multiple source nodes and sink nodes. At last, we demonstrate that the *Physarum* solver converges to the user-optimized (Wardrop) equilibrium by dynamically updating the costs of links in the network. In addition, convergence of the developed algorithm is proved. Numerical examples are used to demonstrate the efficiency of the proposed algorithm. The superiority of the proposed algorithm is demonstrated in comparison with several other algorithms, including the Frank-Wolfe algorithm, conjugate Frank-Wolfe algorithm, biconjugate Frank-Wolfe algorithm, and gradient projection algorithm.**

***Index Terms*—Network optimization, *Physarum*, traffic assignment, user equilibrium.**

I. Introduction

EVER growing demands on transportation and communication networks cause increasingly serious congestions, resulting in substantial losses to businesses all over the world. Various measures have been implemented by governments to mitigate the congestion in existing networks [1]–[3], e.g., congestion toll [4], expansion of the lanes [5], etc. Among them, an effective transportation network design is a good initial strategy that will reduce pressure on future mitigation efforts.

To design a highly efficient network, it is important to account for the dynamics of drivers' behavior. Typically, in a transportation network, drivers noncooperatively seek to minimize their own cost by taking the path with the least perceived cost from the origin to the destination. Typically, the link cost is a monotonically increasing function of the traffic flow. When the cost increases on some paths, drivers switch to other alternate paths. Ultimately, no drivers can reduce his traveling cost by shifting to any other route. A set of link flows in the network is assumed to be at equilibrium if two conditions are satisfied for each origin-destination (OD) pair (s, t): 1) if two or more routes between s and t are traveled by the users, then the cost for each of these routes must be the same and 2) there is no alternate unused route between s and t with less cost than the routes that are traveled. Thus, it is worthwhile to investigate the traffic assignment at the equilibrium state because it plays an essential role in guiding effective design of the transportation networks.

The traffic assignment problem was originally formulated in the 1950s, and various solution algorithms have been proposed since then [6]–[12]. These algorithms can be classified into three categories according to the way that the solution is represented: 1) link-based; 2) path-based; and 3) bush-based. One of the most well-known link-based algorithms is the Frank-Wolfe (FW) algorithm [13]. In each iteration of the FW algorithm, we minimize a linearized objective function by allocating all the traffic to the minimum cost path, in which link costs are determined by the link flows in the current solution for the main problem. But the FW algorithm has been criticized because it tails very badly [11], i.e., the solution bogs down into an endless creep, and never reaches the minimum of the objective function. Many attempts have been made to improve the efficiency of the FW algorithm [14], [15], such as restricted simplicial decomposition [16], conjugate Frank-Wolfe, and biconjugate Frank-Wolfe (BFW) methods [17].

Different from the link-based algorithms, the path-based algorithms decompose the traffic assignment problem into a series of subproblems, and approach the equilibrium solution asymptotically by sequentially shifting the path flows [18], [19]. The first path-based algorithm was developed by Dafermos [20], and it shifted the flow from the path with maximum cost to the path with minimum cost. However, path-based algorithms have been questioned because they need to store all the paths connecting all the OD pairs, which is not realistic for many large-scale networks. To overcome the aforementioned shortcoming, Leventhal *et al.* [21] presented a column generation algorithm for solving a class of nonlinear traffic assignment problems, which only generated the required paths instead of storing all paths joining each OD pair. Later, Jayakrishnan *et al.* [22] developed a gradient projection (GP) method, and it was similar to the path equilibration in the sense that the OD flow was moved to the current shortest path from other nonshortest paths. This algorithm has a parameter α, which is used to determine the amount of flow to be shifted at each step. Jayakrishnan *et al.* [22] recommended to

Manuscript received March 20, 2017; accepted April 4, 2017. This paper was recommended by Associate Editor A. F. S. Gomez. *(Corresponding author: Sankaran Mahadevan.)*

The authors are with the Department of Civil and Environmental Engineering, Vanderbilt University, Nashville, TN 37235 USA (e-mail: sankaran.mahadevan@vanderbilt.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2017.2691666

set it to 1. Chen *et al.* [23] suggested a self-adaptive step size strategy for GP. In this paper, we do not adjust the value of α for each test instance. Instead, we fixed it to a constant value that allows all the test instances to converge.

In recent years, researchers have centered on bush-based algorithms to address the traffic assignment problem [10]. The basic idea is to decompose the problem into a sequence of subproblems, which operate on acyclic subnetworks of the original transportation network [24]. The first bush-based algorithm named origin-based algorithm (OBA) was developed by Bar-Gera [10], which updated the restricting subnetwork and the origin-based link flows. Later, Nie [25] made some refinements to the original OBA. In 2006, Dial [11] presented a novel user-equilibrium traffic assignment algorithm called Algorithm B, which solved the traffic assignment problem by decomposing the problem into a sequence of easy single-origin problems on acyclic subnetworks.

However, the available approaches to the traffic assignment problem still have some limitations and deficiencies. To be specific, FW-based algorithms suffer from expensive computational cost when equilibrium solutions with high precision are required [e.g., when the relative gap (RGAP) is 10^{-6} or less] [25]. Most path-based algorithms for solving the user equilibrium problem need to implement a column generation procedure to generate the required paths. Furthermore, both path-based and bush-based algorithms require different kinds of optimization techniques and approaches to shift the flow from the set of costlier paths to the set of cheaper paths while the optimization in each iteration might consume much more time with the increase of network size.

Last but not least, all the present algorithms for solving the traffic assignment problem are composed of two separate processes: 1) path finding and 2) flow shifting. For example, the FW algorithm starts with an initial feasible link flow pattern generated by an all or nothing assignment (AON). Each link flow is initialized as zero, then the link cost is calculated. For each OD pair, the shortest path between them is found and all the travel demand is allocated to the shortest paths. In this way, the initial link flows, denoted by $\mathbf{y_a^1}$, $\forall a \in A$, where A is the set of links in the traffic network, are generated. After all the path flows are projected on the corresponding links and the link cost is updated, FW further exploits AON and generates another link flow pattern $\mathbf{y_a^2}$. The second step for the FW method is to perform a linear search in the FW direction to shift the flow between $\mathbf{y_a^1}$ and $\mathbf{y_a^2}$. Other algorithms, such as BFW, CFW, and Algorithm B, can be structured in a similar framework. Thus, path finding and flow shift are implemented as separate processes. Can we integrate the two processes into one and implement a simple but efficient method to perform both flow shift and path finding simultaneously?

Recent studies on the biological organism *Physarum* provide us a new insight into this problem. *Physarum polycephalym* is a living substrate and it contains a network of tubes, by means of which nutrients and chemical signals circulate throughout the organism [26]. *Physarum* is able to disassemble and reassemble the tube structure and tube thickness over time in response to the external conditions (availability of food sources), which makes it very useful for studying the dynamic evolution of complex networks [27], [28]. In 2000, Nakagaki *et al.* [26] demonstrated its ability to approximate the shortest path in a labyrinth. Several years later, Tero *et al.* [29] formulated a mathematical model of the adaptive dynamics of a transport network of the slime mold *Physarum polycephalum* to describe its path-finding behavior. In 2010, Tero *et al.* [30] showed that the *Physarum polycephalum* is capable of forming networks of size and complexity comparable to the Tokyo railway system, with similar efficiency, fault tolerance, and overall network length. Since then, *Physarum* has received increasing attention from researchers in different domains. It has been used to address many complex problems, such as network design [31]–[33], system optimization [34]–[36], machine design [37], and route selection [38]–[40].

One of the most promising features in *Physarum* is its adaptability. Specifically, it can adjust its tube thickness to dynamically adapt/react to the changes in the surrounding environment. Such adaptivity inspires us to develop a novel solution to the traffic network equilibrium problem, in which we can update the cost along each link and take advantage of the *Physarum* principle to redistribute the flow in response to the update of the link cost. In the *Physarum* model, since the flow tends to pass through the shortest path, it is reallocated dynamically and automatically once the cost of the links is updated. As time goes on, the network converges to an equilibrium state, which is the so-called Wardrop equilibrium state [41]. In addition, *Physarum* enjoys additional properties when it is applied to address the user equilibrium traffic assignment problem. First of all, the algorithm converges at approximately an exponential rate. Hence, the optimal solution can be found by the *Physarum* algorithm in a very short time. Second, different from the current algorithms, there is no need for the *Physarum* algorithm to perform any optimization, and we only need to solve a set of linear equations. In most cases, the linear equations are sparse, and techniques such as LU decomposition can be used to enhance the performance of the *Physarum* algorithm further.

However, up to now, there is no attempt to employ *Physaum* to deal with the traffic assignment problem. As a result, we are motivated to develop a novel method based on the *Physarum* solver to address the user equilibrium traffic assignment problem. We make three new contributions in this paper. First, the original *Physarum* model is generalized to address the shortest path problem (SPP) in directed networks and asymmetric graphs. Second, the *Physarum* model is extended to solve the graph optimization problem with multiple sources and sinks. Third, a novel strategy is proposed to update the link cost, which makes the *Physarum* model converge to the Wardrop equilibrium state. In addition, we also prove the convergence of Physarum algorithm to the user equilibrium.

The remainder of this paper is structured as follows. In Section II, we introduce the traffic network equilibrium assignment problem and the *Physarum* approach. In Section III, we develop the proposed *Physarum* method for solving the traffic equilibrium assignment problem. In Section IV, several examples are used to demonstrate the proposed method and to compare the performance of the proposed method with other existing approaches. Section V provides concluding remarks.

II. Preliminaries

In this section, we briefly introduce the traffic network equilibrium assignment problem and *Physarum* solver.

A. Traffic Network Equilibrium Assignment Problem

$\mathbb{OD}$	Set of all OD pairs in network G.
a	A link in network G, where $a \in E$.
$d^{s,t}$	Travel demand between origin node s and destination node t, where $(s,t) \in \mathbb{OD}$.
E	Set of links in network G.
f_a	The flow along link a, where $a \in E$.
G	A strongly connected network.
$P^{s,t}$	Set of cycle-free paths connecting s with t, where $(s,t) \in \mathbb{OD}$.
$q_p^{s,t}$	The flow along path p, where $p \in P^{s,t}$.
V	Set of nodes in network G.
$\delta_{ap}^{s,t}$	A binary variable. $\delta_{ap}^{s,t} = 1$ if the link a is a segment of the path p connecting s with t. Otherwise, $\delta_{ap}^{s,t} = 0$, where $(s,t) \in \mathbb{OD}, p \in P^{s,t}$, and $a \in E$.
(s,t)	OD pair (s,t), where $(s,t) \in \mathbb{OD}$.

Consider a strongly connected network $G(V,E)$, where V and E are sets of nodes and links, respectively. Let $\mathbb{OD}$ be all the OD pairs in network G, for which travel demand $d^{s,t}$ is generated, where $(s,t) \in \mathbb{OD}$. Suppose $q_p^{s,t}$ represents the path flow originated at node s and destined to node t, then we have

$$\sum_{p \in P^{s,t}} q_p^{s,t} = d^{s,t}, \ \forall (s,t) \in \mathbb{OD} \tag{1}$$

where $P^{s,t}$ is a set of cycle-free paths connecting s with t. All path flows must be non-negative to guarantee a meaningful solution

$$q_p^{s,t} \geq 0, \ \forall p \in P^{s,t}, \ \forall (s,t) \in \mathbb{OD}. \tag{2}$$

Let f_a denote the flow along the link a. Then the total flow on the link a is the sum of all paths that include the link

$$f_a = \sum_{(s,t) \in \mathbb{OD}} \sum_{p \in P^{s,t}} q_p^{s,t} \delta_{ap}^{s,t}, \ \forall a \in E \tag{3}$$

where $\delta_{ap}^{s,t} = 1$ if the link a is a segment of the path p connecting s with t. Otherwise, $\delta_{ap}^{s,t} = 0$.

In a transportation network, each user noncooperatively seeks to minimize his/her own cost by taking the path with least perceived cost from their origin to destination. The network is said to be in equilibrium if no user can reduce his/her cost by shifting to other alternative routes. The traffic equilibrium assignment problem can be mathematically formulated as

$$\begin{aligned} \text{Min} \quad & z(x) = \sum_{a \in E} \int_0^{f_a} t_a(x) dx \\ \text{s.t.} \quad & f_a = \sum_{(s,t) \in \mathbb{OD}} \sum_{p \in P^{s,t}} q_p^{s,t} \delta_{ap}^{s,t}, \ \forall a \in E \\ & \sum_{p \in P^{s,t}} q_p^{s,t} = d^{s,t}, \ \forall (s,t) \in \mathbb{OD} \\ & q_p^{s,t} \geq 0, \ \forall p \in P^{s,t}, \forall (s,t) \in \mathbb{OD} \end{aligned} \tag{4}$$

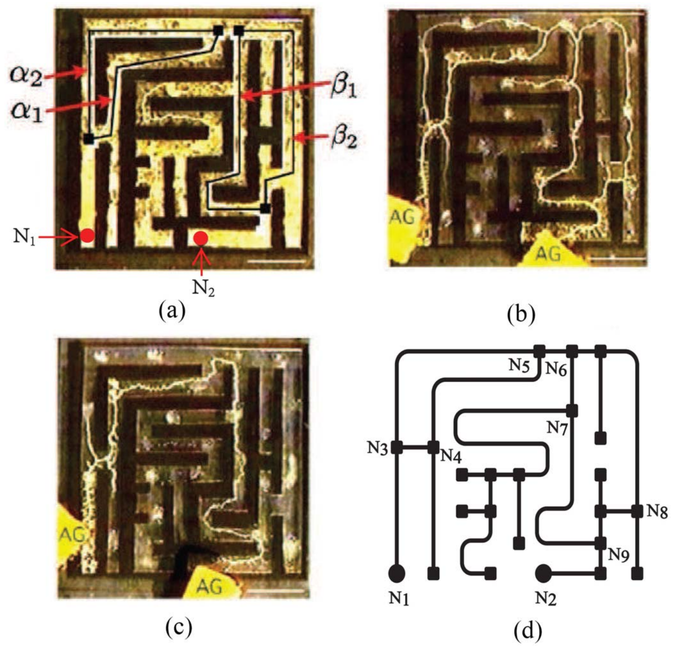


Fig. 1. (a)–(d) illustrates *Physarum*'s maze-solving process. (a) Initial state: maze is filled by *Physarum*. (b) Intermediate state: backbone path is formulated to connect the two food sources. (c) Path spanning the minimum length between the nutrient-containing agar blocks emerges. (d) Graphical representation of the maze: source node N_1 and the sink node N_2 are indicated by solid circles and other nodes are shown by solid squares [29]. The video (https://www.youtube.com/watch?v=czk4xgdhdY4, 2010) shows the experiment.

where z is the objective function, f_a denotes the total traffic flow on link a, $t_a(x)$ is a cost function of link a that is convex and monotonically increasing. $q_{s,t}$ represents the total traffic demand from s to t, $f_p^{s,t}$ denotes the flow on path p between origin node s and destination node t.

B. Physarum Approach

In 2000, Nakagaki *et al.* [26] demonstrated that the plasmodium of the slime mold *Physarum polycephalum* approximates the minimum-length path between two points in a given labyrinth. Before describing the maze-solving process of *Physarum*, we first recall some basic properties of *Physarum polycephalum*. *Physarum* is a large amoeba-like cell, and it consists of a dendritic network of tube-like structures. One of the most important features in *Physarum* is to adapt its shape when it crawls over a plain agar gel. If the food is placed at two different points, *Physarum* formulates the pseudopodia to connect the two points.

Fig. 1 illustrates *Physarum*'s maze-solving process. Initially, the maze is filled by *Physarum* as shown in Fig. 1(a). The two solid red circles, labeled as N_1 and N_2, indicate two separated food sources, where agar blocks containing nutrient are placed in N_1 and N_2, respectively. There are four candidate routes to connect the two food sources, namely, α_1, α_2, β_1, and β_2. Fig. 1(b) demonstrates that the backbone network only containing α_1, α_2, β_1, and β_2 is formulated and all the other tubes in the dead end degenerate. Finally, as shown in Fig. 1(c), the path spanning the minimum length between the nutrient-containing agar blocks emerges.

A mathematical model of *Physarum*'s shortest path finding was developed in Tero *et al.* [29]. Here, we adopt this particular maze as an illustration of the mathematical model of *Physarum*. Suppose we represent the maze by a graph, as shown in Fig. 1(d), in which two special nodes N_1 and N_2 are designated as the starting node and ending node. The other nodes are labeled as $N_3, N_4, \ldots$ etc. The edge/link between nodes N_i and N_j is A_{ij}. The parameter Q_{ij} denotes the flux through tube A_{ij} from node N_i to N_j. Then the flux Q_{ij} can be formulated as

$$Q_{ij} = \frac{D_{ij}}{L_{ij}}(p_i - p_j) \tag{5}$$

where p_i is the pressure at node N_i, D_{ij} is a conductivity (or diameter) of a tube A_{ij}, and L_{ij} is its length.

The inflow and outflow must be balanced

$$\sum_{\forall(i,j)\in A} Q_{ij} = 0 \quad (j \neq 1, 2). \tag{6}$$

The following equations hold for the source node N_1 and the sink node N_2:

$$\begin{aligned} \sum_i Q_{i1} + I_0 &= 0 \\ \sum_i Q_{i2} - I_0 &= 0 \end{aligned} \tag{7}$$

where I_0 is the flux from the source node. We assume I_0 is constant.

To model the adaptive behavior of the slime mold, we assume that the conductivity D_{ij} changes over time according to the flux Q_{ij}. Thus, the evolution process of $D_{ij}(t)$ can be described as follows:

$$\frac{d}{dt}D_{ij} = y(|Q_{ij}|) - \gamma D_{ij} \tag{8}$$

where γ is the decay rate of the tube. The equation implies that the conductivity tends to vanish if there is no flux along the edge, while it is enhanced by the flux. It is natural to assume that y is a monotonically increasing continuous function satisfying $y(0) = 0$.

Then the network Poisson equation for the pressure is derived from (5)–(7) as follows:

$$\sum_i \frac{D_{ij}}{L_{ij}}(p_i - p_j) = \begin{cases} -1 & \text{for } j = 1 \\ +1 & \text{for } j = 2 \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

By setting $p_2 = 0$ as the basic pressure level, all p_i's can be determined uniquely by solving the equation system (9), and each $Q_{ij} = [D_{ij}/L_{ij}](p_i - p_j)$ is also obtained.

In this paper, we assume that $y(Q_{ij}) = |Q_{ij}|$ because *Physarum* can always converge to the shortest path regardless of the initial distribution of conductivities when $y(|Q_{ij}|) = |Q_{ij}|$, and $\gamma = 1$ [29]. With the flux calculated, the conductivity (10) is used instead of (8), adopting the functional form $y(Q) = |Q|$

$$\frac{D_{ij}^{h+1} - D_{ij}^{h}}{\delta t} = \left|Q_{ij}^{h}\right| - D_{ij}^{h+1} \tag{10}$$

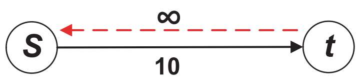


Fig. 2. Directed edge.

where δt is a time mesh size and the upper index h indicates a time step.

In fact, the above equations model a dynamic evolution system, where the conductivity D_{ij} and flow Q_{ij} are functions of time t. The conductivity D_{ij} evolves according to the adaptation equation (10), where variables Q_{ij} and p_i are determined by solving the network Poisson equation in (9) characterized by the value of D_{ij} and L_{ij} at each moment. In equilibrium ($D_{ij}^{h+1} = D_{ij}^{h}$), the conductivity along each link is equal to its flow. In nonequilibrium, the conductivity D_{ij} increases or decreases if the absolute value of the flow is larger than or smaller than the conductivity, respectively.

During the evolution of the system, some links grow or remain while others disappear. The system solves the SPP while the conductivities of the edges on the shortest path converge to one, and the conductivities on the edges outside the shortest path converge to zero. In 2012, Bonifaci *et al.* [42], [43] proved that the mass of *Physarum* will eventually converge to the shortest N_1–N_2 path of the network, independently of the structure of the network or of the initial mass distribution.

III. Efficient *Physarum* Method for the Traffic Equilibrium Assignment

In this section, we develop a *Physarum*-inspired method for the traffic equilibrium assignment problem. Our contributions are three-fold. First, the original *Physarum* model is generalized to address the SPP in directed networks and asymmetric graphs. Second, the *Physarum* model is extended to solve the graph optimization problem with multiple sources and sinks. Third, a novel strategy is proposed to update the link cost so that we can approach the traffic flow assignment at the user equilibria.

A. Shortest Path Problem in Directed Networks

The original *Physarum* model can only handle the SPP in undirected networks according to [29], [44], and [45]. However, in reality, a lot of networks are directed; for example, in many traffic networks, there are many one-way roads. In these situations, the original model is not capable of finding the shortest path. Therefore, we need to extend the *Physarum* model to directed networks.

In a directed network G, suppose L denotes the set of link weights. Given a source node s and a sink node t, the directed SPP is "to find a path from s to t, which only consists of directed edges, with the minimum sum of weights on the edges."

From the mathematical point of view, a directed edge is a special case of a bidirectional link. To be specific, Fig. 2 shows one edge. We assume its orientation is $s \rightarrow t$ with the weight of 10. Whereas, there is no edge from t to s. When this edge is represented in the adjacent matrix, we denote its length as: $L_{st} = 10, L_{ts} = +\infty$. Intuitively, we abstract a directed edge using a bidirectional link, while the weight of the link opposite to the orientation is infinity, see L_{ts}.

In this paper, we adopt the same idea and apply it to the *Physarum* model. Each link (i, j) is characterized by two independent variables L_{ij} and L_{ji}, where L_{ij} denotes the length of the link originating from node i and ending at node j, and L_{ji} represents the length of the link starting from node j to i. As a result, the network Poisson equation formulated in (9) can be updated as

$$\sum_i \left(\frac{D_{ij}}{L_{ij}} + \frac{D_{ji}}{L_{ji}}\right)(p_i - p_j) = \begin{cases} -1 & \text{for } j = 1 \\ +1 & \text{for } j = 2 \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

where D_{ij} is the conductivity of link A_{ij}.

With the refined network Poisson equation (11), we are able to capture the direction of each edge. If there is only a directed link from node i to j, then $L_{ji} = \infty$, $[D_{ji}/L_{ji}] = 0$, and the equation degenerates to the same form as (9), and vice versa. If the link is bidirectional, then $[D_{ij}/L_{ij}] = [D_{ji}/L_{ji}]$, and we can update (9) as

$$\sum_i \frac{2 * D_{ij}}{L_{ij}}(p_i - p_j) = \begin{cases} -1 & \text{for } j = 1 \\ +1 & \text{for } j = 2 \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

The only difference between (9) and (12) is the constant term 2. Since the system evolves according to (10), the introduction of the constant term has no effect on its convergence. According to the evolution (10), the convergence condition still holds when $D_{ij} = Q_{ij}$. Since we bring the constant term, the pressure of each node p'_i shrinks to half of its original value p_i calculated from (9). In the following iterations, the flow along each link shrinks by half because $Q_{ij} = [D_{ij}/L_{ij}](p'_i - p'_j) = [1/2][D_{ij}/L_{ij}](p_i - p_j)$. When the system converges, the conductivities of the edges forming the shortest path converge to 0.5 while the conductivities of the edges outside the shortest path converge to 0.

In addition, the *Physarum* model can be readily represented by an electrical network. In an electrical network, every vertex i has a potential p_i (in the *Physarum*, it models hydrostatic pressure). The resistance R_{ij} is $R_{ij} = [L_{ij}/D_{ij}]$. Let γ_{ij} be the resulting current over the edge (i, j), then we have

$$\gamma_{ij} = \frac{U_{ij}}{R_{ij}} = \frac{p_i - p_j}{\frac{L_{ij}}{D_{ij}}} = \frac{D_{ij}}{L_{ij}}(p_i - p_j) = Q_{ij} \tag{13}$$

which U_{ij} is the potential difference between i and j. It is observed that (13) exactly models the protoplasmic flow across tubes.

In an electrical network, if there is a potential difference in an edge (i, j) from i to j, then an electrical current γ_{ij} will flow from i to j according to Ohm's law. Consider the example shown in Fig. 3, we aim to determine the shortest path from node s to node t. Suppose the lengths of all the links are 1, and s and t are the source and sink nodes. There are two alternate paths to connect s with t : $s \to 1 \to 3 \to t$, and $s \to 2 \to 4 \to t$. Due to the symmetry of the network, the pressure of each node can be ranked as: $p_s > p_1 = p_2 > p_3 = p_4 > p_t$.

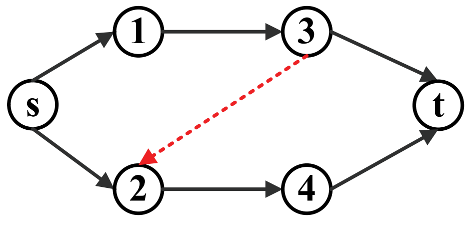


Fig. 3. Simple network.

Now, we add another directed link $(3, 2)$ to the network. Since $p_2 > p_3$, the current tends to flow from nodes 2 to 3, which is opposite to the link's direction. To resolve this problem, we implement a checking procedure to guarantee the flow follows the link's orientation. Specifically, we record the direction of each link. If there is a directed edge from node i to node j, and the pressure p_j at node j calculated from (12) is larger than the pressure p_i at node i, the flux tends to flow from node j to node i, which is opposite to the link's orientation. In this case, since $Q_{ij} = [D_{ij}/L_{ij}](p_i - p_j) < 0$, which means the flow moves from node j to i, we let the flow Q_{ij} to be zero. In other words, we forbid the flow through the network when its direction is opposite to the link's orientation.

B. *Physarum* Solver in Multisource Multisink Network

In a transportation network, there might be multiple source-sink pairs, whereas the original *Physarum* model is limited to a network with only one source node and one sink node. As a result, we modify (12) as follows to address the traffic assignment problem with multiple sources and sinks:

$$\sum_i \left(\frac{D_{ij}}{L_{ij}} + \frac{D_{ji}}{L_{ji}}\right)(p_i - p_j) = \begin{cases} -d^{s,t}, & \forall (s, t) \in \mathbb{OD} \\ +d^{s,t}, & \forall (s, t) \in \mathbb{OD} \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where $\mathbb{OD}$ denotes the set of all OD pairs, and $d^{s,t}$ denotes the travel demand between the origin node s and destination node t.

Given multiple origin and destination nodes, the proposed *Physarum* algorithm chooses the shortest path to transport the demand. Equation (14) guarantees that the flow is transported along the shortest paths, which has been proved by Straszak and Vishnoi [33].

The most time-consuming part in the *Physarum* algorithm is to solve the linear system defined in (14). Fortunately, the linear system is very special, it is a Laplacian system. Specifically, for the given graph $G = (V, E)$, let $B = B_G \in \mathbb{R}^{V \times E}$ denote the incidence matrix of G, which is given by

$$B_{ia} := \begin{cases} -1 & \text{if } i \text{ is the head of arc } a \\ +1 & \text{if } i \text{ is the tail of arc } a \\ 0 & \text{if } i \text{ is not incident with } a. \end{cases}$$

Then the network Poisson equation is equivalent to the following weighted graph Laplacian:

$$L(D) \overset{\text{def}}{=} BGB^T \tag{15}$$

where $G \overset{\text{def}}{=} \text{diag}(L/D) \in {}^{E \times E}$ is the diagonal matrix with the conductance vector L/D along the diagonal, $L(D)$ is the graph

Laplacian. Then the network Poisson's equation amounts to finding a solution p to the discrete Neumann problem [46]

$$L(D)p = d \tag{16}$$

where d is the demand matrix.

In the past decades, researchers have developed a solver that runs in $O(m \log n)$ time for the Laplacian system defined in (16), where m is the number of edges, and n is the number of nodes in the network G [47], [48]. Reference [49] introduces the techniques developed for solving Laplacian systems, which can be implemented to improve the performance of the proposed algorithm further. By structuring the linear system in a Laplacian framework, we are able to reduce the time complexity for solving linear equations from $O(n^3)$ to nearly linear time $O(m)$.

C. Physarum Solver for Traffic Equilibrium Assignment

In the traffic network, link cost is a function of the traffic flow. Every time, the flow volume on a specific link changes, the link cost needs to be updated accordingly, which in turn makes the traffic flow be redistributed in the updated network. The process continues until the system converges to the user equilibrium.

The *Physarum* algorithm enjoys several nice properties when applied to the traffic assignment problem. First of all, the network Poisson equation shown in (14) makes the *Physarum* solver capable of redistributing the flow in accordance with the updated link cost L_{ij}. Every time the cost is updated, the *Physarum* solver adapts itself to redistribute the flow in a way such that the selected paths are optimal after the link cost is updated.

Another unique characteristic in *Physarum* is that the flow is continuous during the iterations, which makes it different from the classical shortest path finding algorithms, such as the Dijkstra algorithm [50], label setting algorithm [51], etc. Specifically, the Dijkstra algorithm performs the optimization in a discrete manner by iterating over all the nodes one by one. When the costs on some links related to the visited node happen to change, the Dijkstra algorithm needs to revisit those nodes and reconstruct the path again. Moreover, in the Dijkstra algorithm, each link is only associated with one criterion—length—and there is no equivalent attribute like "flow" in our *Physarum* model reacting to the change of the link cost. As a result, many classical algorithms for the traffic equilibrium problem must have two separate processes: 1) path finding and 2) flow shift. In contrast, our *Physarum* algorithm is very straightforward; once the link cost L is updated, with the help of (14), the flow is redistributed and reallocated dynamically in the next iteration.

Since the cost function of shipping Q_{ij} units along link (i, j) is denoted as $t_{ij}(Q_{ij})$, we propose the following equation to update the cost of the link (i, j):

$$L_{ij}^{h+1} = \frac{L_{ij}^{h} + t_{ij}\left(Q_{ij}^{h+1}\right)}{2} \tag{17}$$

where Q_{ij}^{h+1} represents the flow along link (i, j) at the $h+1_{\text{th}}$ iteration, L_{ij}^{h+1} and L_{ij}^{h} denote the cost on the link at the $h+1_{\text{th}}$

Algorithm 1 *Physarum* Solver in Traffic Network Equilibrium Assignment Problem $(L, n, \mathbb{OD})$

```
1:  // $n$ is the size of the network;
2:  // $\mathbb{OD}$ is the set of all OD pairs in the network;
3:  // $L_{ij}$ is the length of the link connecting node $i$ with node $j$;
    $D_{ij} \leftarrow (0, 1]$ $(\forall i, j = 1, 2, \ldots, n)$;
    $Q_{ij} \leftarrow 0$ $(\forall i, j = 1, 2, \ldots, n)$;
    $p_i \leftarrow 0$ $(\forall i = 1, 2, \ldots, n)$;
4:  repeat
5:      Calculate the pressure associated with each node according to (14)
        $\sum_i \left(\frac{D_{ij}}{L_{ij}} + \frac{D_{ji}}{L_{ji}}\right)(p_i - p_j) = \begin{cases} -d^{s,t}, & \forall (s,t) \in \mathbb{OD}, \\ +d^{s,t}, & \forall (s,t) \in \mathbb{OD}, \\ 0, & \text{otherwise.} \end{cases}$
6:      $Q_{ij} \leftarrow D_{ij} \times (p_i - p_j) / L_{ij}$ // Using Eq. (5);
7:      $D_{ij} \leftarrow Q_{ij} + D_{ij}$ // Using Eq. (10)
8:      Update the cost on each link;
9:      for $i = 1{:}n$ do
10:         for $j = 1{:}n$ do
11:             $L_{ij} = \frac{L_{ij} + t_{ij}(Q_{ij})}{2}$
12:         end for
13:     end for
14: until the required RGAP is met
```

and h_{th} iteration, respectively. The composite cost function L_{ij}^{h+1} combines the current cost L_{ij}^{h} with the cost in the next iteration denoted by $t_{ij}(Q_{ij}^{h+1})$. The composite cost function guides the algorithm to allocate the flow in a way that balances the cost and flow.

The cost function defined in (17) has several benefits. Specifically, $t_{ij}(Q_{ij}^{h+1})$ denotes the cost on link (i, j) at the $h+1$ iteration provided that Q_{ij}^{h+1} is allocated to link (i, j). Thus, we update the link cost L_{ij}^{h+1} in the way that combines the current link cost L_{ij}^{h} and the future link cost $t_{ij}(Q_{ij}^{h+1})$, which, in turn, affects the flow distribution further in the next iteration according to (14). When the two costs are the same ($L_{ij}^{h} = t_{ij}(Q_{ij}^{h+1})$), the algorithm converges. The test problems in Section IV demonstrate the convergence and correctness of the proposed algorithm.

The general flow of the proposed algorithm is shown in Algorithm 1. There are several possible stopping criteria in Algorithm 1, such as the maximum number of iterations is reached, flux through each tube remains unchanged, etc. The algorithm described in this paper stops when its solution to the traffic assignment problem satisfies the required precision.

D. Convergence of the Proposed Method

In this section, we prove that the developed *Physarum* algorithm converges to the optimal solution that minimizes the objective function (4).

Lemma 1: When the developed Physarum algorithm converges, the traveling time L_{st} among any OD pair (s, t) is a constant.

Proof: As indicated in [42], $D_{ij}^{h+1} = D_{ij}^{h}$ when the *Physarum* algorithm converges.

From (10), it holds that $D_{ij}^{h} = |Q_{ij}^{h}|$. Thus, we have

$$D_{ij}^{h} = \begin{cases} Q_{ij}^{h}, & \text{if } Q_{ij}^{h} \geq 0 \\ -Q_{ij}^{h}, & \text{if } Q_{ij}^{h} < 0. \end{cases} \tag{18}$$

1) If $Q_{ij}^{h} \geq 0$, since D_{ij} and L_{ij} are nonnegative variables, we derive $p_i \geq p_j$ from (5). Besides, since $D_{ij}^{h} = Q_{ij}^{h}$, (5) further implies $L_{ij} = p_i - p_j$.
2) If $Q_{ij}^{h} < 0$, we have $p_i < p_j$. By substituting $D_{ij}^{h} = -Q_{ij}^{h}$ into (5), we have $L_{ij} = p_j - p_i$.

In summary, it holds that $L_{ij} = |p_i - p_j|$, where p_i and p_j are derived from (14). The linear system defined in (14) determines the node potential uniquely. In other words, the potential at each node is a constant, so we have $L_{ij}^{h+1} = L_{ij}^{h}$ when $h \to \infty$. Likewise, for any OD pair (s, t), L_{st} is a constant. ■

Lemma 2: When the developed Physarum algorithm converges, the travel time along any link is equal to the cost derived from the cost function.

Proof: From Lemma 1, we have $L_{ij}^{h+1} = L_{ij}^{h}$. By substituting it into (17), we have $L_{ij}^{h+1} = t_{ij}(Q_{ij}^{h+1})$ when $h \to \infty$. ■

Lemma 3: The optimal solution of the mathematical minimization program formulated in (4) is equivalent to the user equilibrium conditions.

Proof: Sheffi [52, pp. 63–66] proved the equivalence between the equilibrium conditions and the optimal solution to the minimization program of the traffic assignment problem. Here, we do not repeat the proof for the sake of brevity. Thus, the solution satisfying the equilibrium conditions is the one that minimizes the objective function defined in (4). ■

Lemma 4: The objective function formulated in (4) has a unique minimum.

Proof: The Hessian matrix of the objective function (4) can be calculated by using a representative term of the matrix. The derivative is first computed with respect to the flow on the mth and nth link, respectively

$$\frac{\partial z(x)}{\partial x_m} = t_m(x_m) \tag{19}$$

and the second derivative is

$$\frac{\partial^2 z(x)}{\partial \mathrm{x_m} \partial \mathrm{x_n}} = \frac{\partial t_m(x_m)}{\partial x_n} = \begin{cases} \frac{dt_n(x_n)}{dx_n}, & \text{for } m = n \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

Obviously, all the off-diagonal elements of the Hessian matrix, $\nabla^2 z(x)$, are zero, and all the diagonal elements are given by $[dt_a(x_a)/dx_a]$. Thus, we have

$$\nabla^2 z(x) = \begin{bmatrix} \frac{dt_1(x_1)}{dx_1} & 0 & 0 & \cdots \\ 0 & \frac{dt_2(x_2)}{dx_2} & 0 & \cdots \\ 0 & 0 & \ddots & \cdots \\ \vdots & \vdots & \vdots & \frac{dt_a(x_a)}{dx_a} \end{bmatrix}. \tag{21}$$

As all entries in the diagonal are strictly positive, this matrix is positive definite. Thus, the objective function (4) is strictly convex. In other words, the traffic assignment problem has a unique minimum. ■

TABLE I
BASIC CHARACTERISTICS OF TEST NETWORKS

Network	Nodes	Links	O–D pairs
test network 1	13	19	10
test network 2	25	40	10
test network 3	24	76	6
test network 4	416	914	7
test network 5	933	2,950	12

From Lemmas 1 and 2, it entails that all the used paths have the same cost. Thus, the first condition of the user equilibrium is satisfied.

As indicated in [42], the *Physarum* model always converge to the shortest path, and the conductivity along the edges outside the shortest path converges to zero. Hence, there is no flow along the nonoptimal paths because their traveling time is inferior to that of the shortest paths. Thus, the second condition of the user equilibrium is met. From Lemmas 3 and 4, we know that the solution satisfying the two user equilibrium conditions is equivalent to the one that minimizes the program (4). Hence, the proposed *Physarum* model converges to user equilibrium.

E. Measure of Solution Quality

In this paper, we use the RGAP to measure the convergence

$$\mathrm{RGAP} = 1 - \frac{\sum_{s \in \mathbb{O}} \sum_{t \in \mathbb{D}} d^{st} \cdot p_{\min}^{st}}{\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot t_{ij}} \tag{22}$$

where d^{st} represents the travel demand between node s and t, $p_{\min}^{st}$ denotes the shortest path of O–D pair (s, t), f_{ij} and t_{ij} denote the flow and travel time along the link (i, j) when the algorithm terminates.

IV. Test Problems

In this section, several test networks are used to demonstrate the efficiency of the proposed algorithm, and compare its performance against those of the FW, BFW, CFW, and GP algorithms using several real-size networks, such as the Chicago Sketch and Anaheim networks, etc. All the five algorithms are implemented using C++. In FW, BFW, and CFW, we implement linear search to balance the flow. All tests are performed in a Windows 7 computer with Intel Core i5-2510 CPU, 4 Core, 2.5 GHz; 4 GB RAM. The RGAP is set as 10^{-5} for the fist four examples, 10^{-4} for the last example. The basic characteristics of the test networks are presented in Table I.

We adopt the following cost function developed by the U.S. Bureau of Public Roads (BPR) for all the links [53]

$$t_{ij} = \alpha_{ij}\left(1 + 0.15\left(\frac{v_{ij}}{C_{ij}}\right)^4\right) \tag{23}$$

where t_{ij}, α_{ij}, v_{ij}, and C_{ij} denote the travel time (cost), free-flow travel time, flow and capacity along link (i, j), respectively.

A. Example 1

The network shown in Fig. 4 is adapted from [54], and it has 13 nodes, 19 links, and ten OD pairs. The OD demands,

TABLE II
LINK CHARACTERISTICS FOR NGUYEN-DUPUIS'S 13-NODE NETWORK SHOWN IN FIG. 4 [54]

Link	Free-flow travel time (min/trip)	Capacity (veh/h)	Link	Free-flow travel time (min/trip)	Capacity (veh/h)
1	7	300	11	9	500
2	9	200	12	10	550
3	9	200	13	9	200
4	12	200	14	6	400
5	3	350	15	9	300
6	9	400	16	8	300
7	5	500	17	7	200
8	13	250	18	14	300
9	5	250	19	11	200
10	9	300			

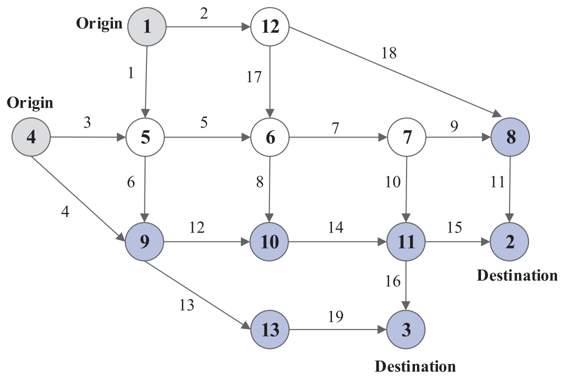


Fig. 4. Test network 1: Nguyen-Dupuis's 13-node network [54], the gray colored circles represent the origins, and the blue colored circles denote the destinations.

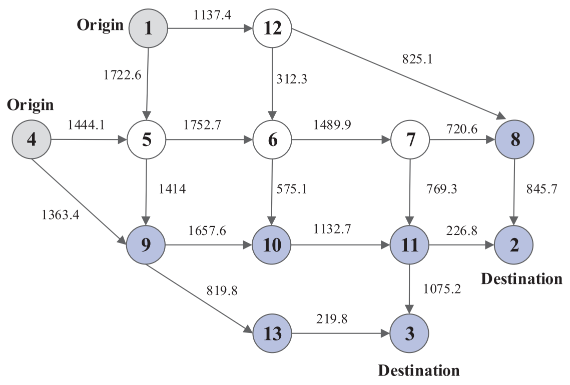


Fig. 5. Optimal solution for the traffic assignment in test network 1.

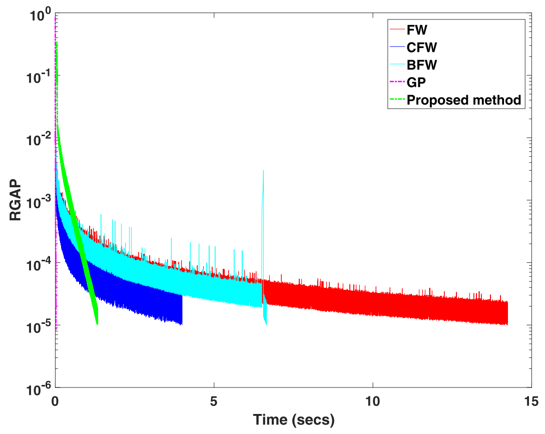


Fig. 6. Computational effort versus accuracy in test network 1.

in vehicles per hour, are shown as

$$q^{1,2} = 660,\ q^{1,3} = 800,\ q^{1,10} = 800,\ q^{1,11} = 600$$
$$q^{4,2} = 412.5,\ q^{4,3} = 495,\ q^{4,8} = 700,\ q^{4,9} = 300$$
$$q^{4,10} = 300,\ q^{4,13} = 600$$

and the link characteristics are shown in Table II.

The optimal solution found by the proposed Physarum algorithm is shown in Fig. 5, and the result is consistent with that of the other four algorithms. From the flow along each link indicated in Fig. 5, we can compute the travel time along each link according to the BPR function defined in (23). With the calculated travel cost, we compute the travel time along the alternate paths that connect the same OD pair. Specifically, the cost along the path $1 \rightarrow 12 \rightarrow 8 \rightarrow 2$ is $1421.1 + 134.1390 + 20.0464 = 1575.3$, and the traveling time is the same as that of the other two paths, namely, $1 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 2$, and $1 \rightarrow 12 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 2$. Likewise, this also applies to the paths that connect other OD pairs. Obviously, all the used paths connecting the same OD pair have the same travel time.

Fig. 6 illustrates the computational comparisons between the proposed method and the other four methods. Among all the algorithms, GP consumes the least amount of time to converge to the predefined precision, and the proposed algorithm is the second fastest algorithm among them. Since this is a tiny example, there is no explicit difference in the running time of GP and the developed algorithm. In the subsequent examples, we will demonstrate the superiority of the proposed algorithm against GP in searching for the equilibrium solution. Fig. 6 also reveals the FW algorithm's quick start/never-end feature. In fewer than 2 s, FW reduced the RGAP from 0.89 to 10^{-3}, a difference of 0.899. Yet in 14 s, it could reduce RGAP further by only $9 * 10^{-4}$. FW spent nearly seven times as long to achieve less than one-hundredth of the reduction. Besides, we also observe the zigzag pattern in the FW-based algorithms. This happens because the search direction is perpendicular to the gradient of the objective function when the algorithm approaches the equilibrium solution [17]. But GP and the proposed algorithm does not have this issue, and such characteristics make them more suitable to address the traffic assignment problems in practical applications.

B. Example 2

The network shown in Fig. 7 has 40 two-way links and ten OD pairs. The link cost is asymmetric in this network. The nodes in squares represent the source and sink nodes, and all the remaining 21 nodes are transshipment nodes. The numbers along each link represent the link capacity and free flow travel time, respectively. For example, the free flow travel time on link $(1, 2)$ is 3.72 and the link capacity is 300. Suppose the travel demand matrix is given as

$$q^{1,22} = 700,\ q^{1,25} = 500,\ q^{2,22} = 800,\ q^{5,24} = 600$$
$$q^{5,25} = 800,\ q^{6,10} = 800,\ q^{6,24} = 2800,\ q^{11,22} = 800$$
$$q^{14,22} = 800,\ q^{15,20} = 800.$$

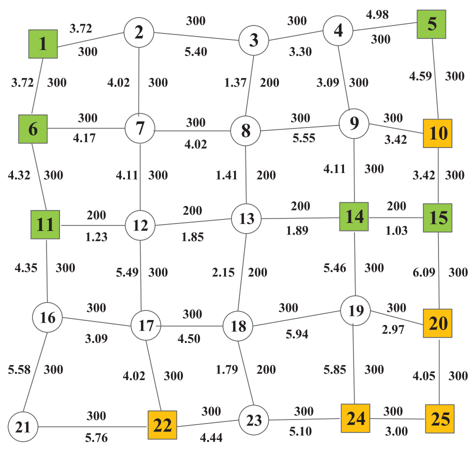


Fig. 7. Test network 2: transportation network with 25 nodes, the green colored squares represent the origins, and the yellow colored squares denote the destinations.

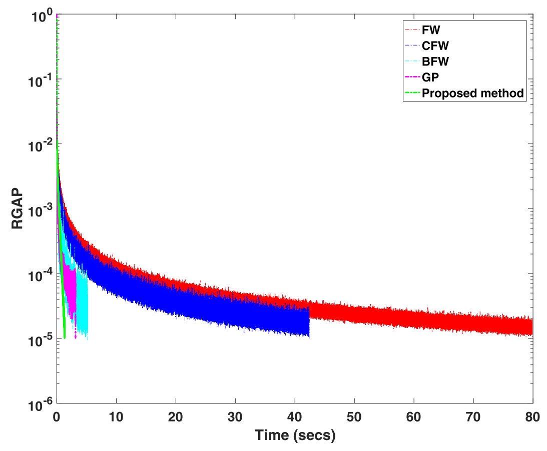


Fig. 8. Computational effort versus accuracy in test network 2.

Fig. 8 shows the comparison results for a target RGAP of 10^{-5}. Among FW-based algorithms, BFW demonstrates the best performance, followed by CFW. All the FW-based algorithms suffer from the problem that the search direction is perpendicular to the gradient of the objective function when the level of RGAP reaches 10^{-4}. The proposed method runs much faster than the FW, CFW, BFW, and GP algorithms, and it converges to the equilibrium solution in 2 s. As can be observed, the proposed algorithm outperforms GP in searching for the equilibrium solution. While GP takes 3.2 s to converge, the developed algorithm only costs 1.3 s to obtain the equilibrium solution. Besides, GP also shows a zigzag pattern when the value of RGAP is less than 10^{-4}. In contrast, the convergence curve of the developed algorithm is very smooth and it does not suffer from such problems.

C. Example 3

Test network 3 is the classical Sioux Falls network with 24 nodes and 76 links. The travel time function is defined as

$$t_a(x_a) = \alpha_a + \beta_a x_a^4 \tag{24}$$

where x_a denotes the flow on link a.

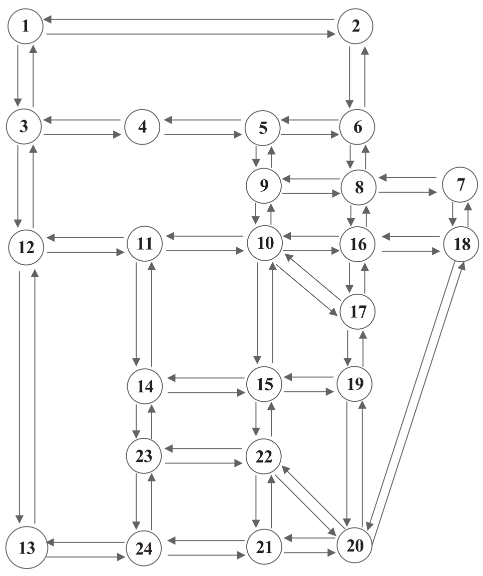


Fig. 9. Test network 3: Sioux Falls network.

TABLE III
PARAMETERS OF THE TRAVEL TIME FUNCTIONS FOR THE NETWORK IN FIG. 9 (α IS GIVEN IN HOURS, β IS GIVEN IN HOURS PER THOUSAND VEHICLES)

Links	Parameters		Links	Parameters	
	α	β		α	β
(1, 2), (2, 1)	0.06	0.00000002	(11, 12), (12, 11)	0.06	0.00001550
(1, 3), (3, 1)	0.04	0.00000002	(11, 14), (14, 11)	0.04	0.00001061
(2, 6), (6, 2)	0.05	0.00001241	(12, 13), (13, 12)	0.03	0.00000001
(3, 4), (4, 3)	0.04	0.00000007	(13, 24), (24, 13)	0.04	0.00000893
(3, 12), (12, 3)	0.04	0.00000002	(14, 15), (15, 14)	0.05	0.00001085
(4, 5), (5, 4)	0.02	0.00000003	(14, 23), (23, 14)	0.04	0.00001020
(4, 11), (11, 4)	0.06	0.00001550	(15, 19), (19, 15)	0.03	0.00000010
(5, 6), (6, 5)	0.04	0.00001001	(15, 22), (22, 15)	0.03	0.00000053
(5, 9), (9, 5)	0.05	0.00000075	(16, 17), (17, 16)	0.02	0.00000401
(6, 8), (8, 6)	0.02	0.00000521	(16, 18), (18, 16)	0.03	0.00000003
(7, 8), (8, 7)	0.03	0.00000119	(17, 19), (19, 17)	0.02	0.00000554
(7, 18), (18, 7)	0.02	0.00000001	(18, 20), (20, 18)	0.04	0.00000002
(8, 9), (9, 8)	0.10	0.00002306	(19, 20), (20, 19)	0.04	0.00000958
(8, 16), (16, 8)	0.05	0.00001157	(20, 21), (21, 20)	0.06	0.00001373
(9, 10), (10, 9)	0.03	0.00000012	(20, 22), (22, 20)	0.05	0.00001130
(10, 11), (11, 10)	0.05	0.00000075	(21, 22), (22, 21)	0.02	0.00000401
(10, 15), (15, 10)	0.06	0.00000027	(21, 24), (24, 21)	0.03	0.00000790
(10, 16), (16, 10)	0.04	0.00001080	(22, 23), (23, 22)	0.04	0.00000960
(10, 17), (17, 10)	0.08	0.00001930	(23, 24), (24, 23)	0.02	0.00000451

The parameters of the travel time function are given in Table III. As can be noted, each link is bi-directional. The cost on the bidirectional link is asymmetric. The travel demand is given as

$$q^{1,20} = 20000,\ q^{1,24} = 40000,\ q^{3,24} = 30000$$
$$q^{4,20} = 20000,\ q^{5,14} = 30000,\ q^{18,20} = 10000.$$

Our goal is to find an optimal way to allocate the flow through the network to satisfy the OD demand as well as balancing the cost of each alternative path in the network. Fig. 10 illustrates the comparison results by different algorithms for a target RGAP of 10^{-5}. Obviously, the proposed *Physarum* algorithm and GP algorithm outperform FW, BFW, and CFW in terms of running time. From Fig. 10, it can be seen that CFW starts outperforming FW when the RGAP is around $8 * 10^{-3}$, whereas BFW starts outperforming the other methods when the RGAP is around 10^{-3}, reaching its full advantage when RGAP is around 10^{-4}. With respect to GP and the proposed algorithm, to make the difference between

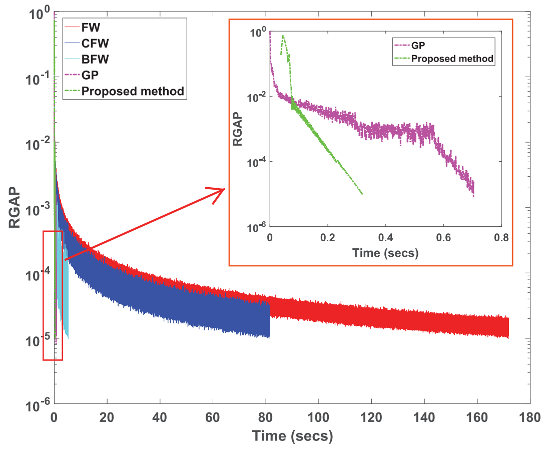


Fig. 10. Computational effort versus accuracy in Sioux-Falls network.

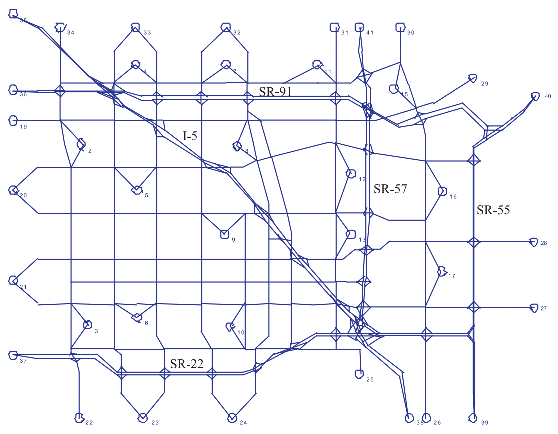


Fig. 11. Anaheim network.

them observable, we attach a figure inside Fig. 10 to compare their computational performance. From the nested figure, it can be seen that the proposed method outperforms GP in searching for the solution.

D. Example 4

Fig. 11 shows the Anaheim network in CA, USA with 461 nodes, and 914 links in this network. The detailed data, such as free flow travel time, the cost function, and the capacity, on these links are available at the Transportation Network Test Problems website (http://www.bgu.ac.il/∼bargera/tntp/), and are not reproduced here.

The travel demand is given as

$$q^{1,416} = 20000, q^{2,400} = 20000, q^{3,410} = 10000$$
$$q^{3,416} = 20000, q^{54,416} = 10000, q^{25,408} = 30000$$
$$q^{40,380} = 5000.$$

The proposed *Physarum* algorithm finds the optimal solution in less than 10 s with a precision of 10^{-5}, as indicated in Fig. 12. In contrast, all the other algorithms consume much more time than the *Physarum* algorithm. CFW and BFW almost have the same trend during the iteration process. Both of them are much faster than the FW algorithm. GP is faster than all the FW-based algorithms, but still slower than the proposed algorithm. It can be noticed that the developed algorithm has clear advantage over all the other four algorithms in identifying the equilibrium solution in the Anaheim network.

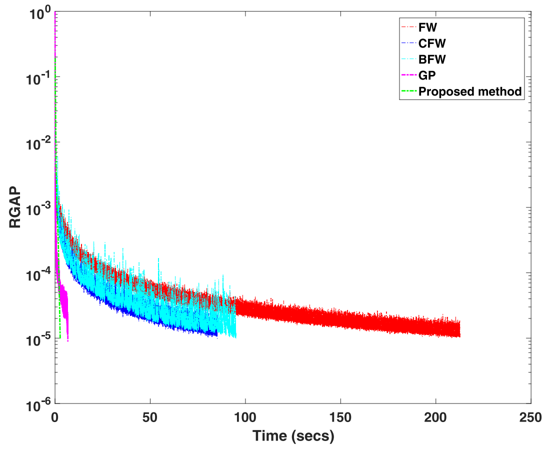


Fig. 12. Comparison between the proposed method and other approaches in Anaheim network.

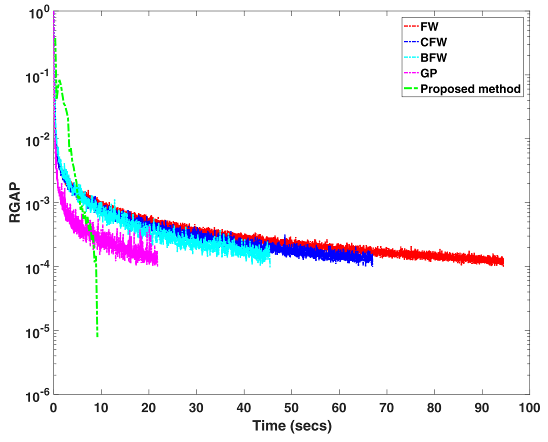


Fig. 13. Comparison between the proposed method and other approaches in Chicago Sketch network.

E. Example 5

In this example, we compare the performance of the five algorithms in solving the user equilibrium in the Chicago Sketch network. There are 933 nodes and 2950 links in the network. The travel demand is given as

$$q^{1,782} = 6000, q^{2,915} = 3000, q^{3,933} = 40000$$
$$q^{12,821} = 3000, q^{14,912} = 4500, q^{16,912} = 5800$$
$$q^{18,933} = 6800, q^{21,908} = 40000, q^{24,745} = 4000$$
$$q^{30,921} = 5000, q^{45,904} = 20000, q^{56,879} = 6000.$$

Fig. 13 illustrates the computational performance of the five algorithms for the Chicago Sketch network measured by the RGAP as a function of running time, both on logarithmic scale. This figure demonstrates the clear advantage of the proposed algorithm over the other four algorithms. In addition, the GP algorithm also shows zigzag pattern when the level of RGAP is less than 10^{-3}. Whereas, the proposed algorithm has no such problem and is more stable. All the FW-based algorithms present similar characteristics as the previous numerical examples.

All the above five examples show that the proposed *Physarum* algorithm is very efficient in addressing the traffic equilibrium assignment problem in terms of both accuracy and the execution time. Five networks with different structures and sizes ranging from nine nodes to 915 nodes were considered. The *Physarum* algorithm is found to have superior performance than the FW, CFW, BFW, and GP algorithms.

V. Conclusion

This paper proposed a novel algorithm—*Physarum* solver—to address the traffic network equilibrium assignment problem. The algorithm converges to the optimal solution by automatically updating the cost on each link and distributes the demand between OD pairs through the network in a way that approaches the equilibrium assignment. Several numerical examples were used to illustrate the proposed method. We verified our approach by comparing the solutions and performance with the FW algorithm, CFW algorithm, BFW algorithm, and GP algorithm. The proposed method is seen to outperform the other methods on several networks with different sizes.

The main contribution of this paper is to extend a *Physarum*-based method to solve the traffic assignment problem, which integrates the path finding and flow redistribution processes into one. Our contributions are threefold: 1) the original *Physarum* model is generalized to solve the SPP in directed networks and asymmetric graphs; 2) the *Physarum* model is extended to address the network optimization problem with multiple sources and sinks; and 3) a composite function is constructed to update the link cost in a way such that the proposed algorithm is able to converge to the solution of traffic assignment problem.

The proposed method is very simple to implement. All the advantages of the proposed method are attributed to *Physarum's* unique features: adaptivity and continuity of the flow. Every time the link cost is updated, the flow gets reallocated through the network according to the network Poisson law, which in turn, updates the link cost again. In summary, the proposed method has the following advantages over the classical approaches.

1) FW, BFW, and CFW all perform AON assignment based on the shortest path finding algorithms, and different strategies are needed to shift the flow through the network to approach the traffic equilibria. The path finding and flow redistribution are implemented as two separate processes in these algorithms whereas the *Physarum*-based method integrates them into one, and solves both the path finding and flow shifting simultaneously.
2) The *Physarum*-based method updates the flow on each link when the link cost changes. The flow gets updated and shifted automatically, guided by the network Poisson flow law. That is where its effectiveness derives from.
3) The FW, BFW, CFW and GP algorithms all need to perform different kinds of optimization to determine how much flow is shifted from the costlier path to the cheaper path. In contrast, the *Physarum* solver does not need any optimization technique. We only need to solve a set of linear equations, which significantly reduces the computational burden compared to the other methods.

Although the *Physarum* algorithm is very promising, it still has some limitations. For example, on the right side of the network Poisson equation shown in (14), the traffic flow at each node is only represented by one variable denoted by d. Suppose there is two-way traffic flow between node A and node B, denoted by $S_{A,B}$ and $S_{B,A}$, it is difficult for the *Physarum* model to address this problem because we only have one variable d to represent the total demand at each node. This results in $d_A = -S_{A,B} + S_{B,A}$, which is different from the realistic scenario. One possible way is to add an intermediate node between A and B. However, such a strategy will increase the problem complexity.

Further research is needed in two different directions. The *Physarum* model needs to be investigated for extension to the generalized traffic assignment problem. It also needs to be studied for applications to more complicated scenarios, such as multiclass, multicriteria traffic network equilibrium assignment, traffic assignment with elastic demand, and the bilevel network design.

Acknowledgment

The authors are grateful to the anonymous reviewers for their constructive suggestions. Their insights and comments greatly improved the presentation of the ideas in this paper.

References

[1] T. Yamasaki and T. Ushio, "An application of a computational ecology model to a routing method in computer networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 1, pp. 99–106, Feb. 2002.

[2] Y. Lou, Y. Yin, and J. A. Laval, "Optimal dynamic pricing strategies for high-occupancy/toll lanes," *Transp. Res. C Emerg. Technol.*, vol. 19, no. 1, pp. 64–74, 2011.

[3] R. Sabella and P. Iovanna, "Self-adaptation in next-generation Internet networks: How to react to traffic changes while respecting QoS?" *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 6, pp. 1218–1229, Dec. 2006.

[4] E. T. Verhoef, "Second-best congestion pricing in general networks. Heuristic algorithms for finding second-best optimal toll levels and toll points," *Transp. Res. B Methodol.*, vol. 36, no. 8, pp. 707–729, 2002.

[5] R. Cervero, "Road expansion, urban growth, and induced travel: A path analysis," *J. Amer. Plan. Assoc.*, vol. 69, no. 2, pp. 145–163, 2003.

[6] L. J. LeBlanc, E. K. Morlok, and W. P. Pierskalla, "An efficient approach to solving the road network equilibrium traffic assignment problem," *Transp. Res.*, vol. 9, no. 5, pp. 309–318, 1975.

[7] S. Nguyen and S. Pallottino, "Equilibrium traffic assignment for large scale transit networks," *Eur. J. Oper. Res.*, vol. 37, no. 2, pp. 176–186, 1988.

[8] A. Chen, J.-S. Oh, D. Park, and W. Recker, "Solving the bicriteria traffic equilibrium problem with variable demand and nonlinear path costs," *Appl. Math. Comput.*, vol. 217, no. 7, pp. 3020–3031, 2010.

[9] H.-J. Huang and Z.-C. Li, "A multiclass, multicriteria logit-based traffic equilibrium assignment model under ATIS," *Eur. J. Oper. Res.*, vol. 176, no. 3, pp. 1464–1477, 2007.

[10] H. Bar-Gera, "Origin-based algorithm for the traffic assignment problem," *Transp. Sci.*, vol. 36, no. 4, pp. 398–417, 2002.

[11] R. B. Dial, "A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration," *Transp. Res. B Methodol.*, vol. 40, no. 10, pp. 917–936, 2006.

[12] M. Patriksson, *The Traffic Assignment Problem: Models and Methods*. New York, NY, USA: Courier Dover, 2015.

[13] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Res. Logistics*, vol. 3, nos. 1–2, pp. 95–110, 1956.

[14] M. Lupi, "Convergence of the Frank–Wolfe algorithm in transportation networks," *Civil Eng. Syst.*, vol. 3, no. 1, pp. 7–15, 1986.

[15] Z. Zhou, A. Brignone, and M. Clarke, "Computational study of alternative methods for static traffic equilibrium assignment," in *Proc. World Conf. Transp. Res. Soc. (WCTRS)*, Lisbon, Portugal, 2010.

[16] D. W. Hearn, S. Lawphongpanich, and J. A. Ventura, "Finiteness in restricted simplicial decomposition," *Oper. Res. Lett.*, vol. 4, no. 3, pp. 125–130, 1985.

[17] M. Mitradjieva and P. O. Lindberg, "The stiff is moving—Conjugate direction Frank–Wolfe methods with applications to traffic assignment," *Transp. Sci.*, vol. 47, no. 2, pp. 280–293, 2013.

[18] M. Florian, I. Constantin, and D. Florian, "A new look at projected gradient method for equilibrium assignment," *Transp. Res. Rec. J. Transp. Res. Board*, vol. 2090, no. 2090, pp. 10–16, 2009.

[19] A. Kumar and S. Peeta, "An improved social pressure algorithm for static deterministic user equilibrium traffic assignment problem," in *Proc. 90th Annu. Meeting Transp. Res. Board*, Washington, DC, USA, 2011.

[20] S. C. Dafermos, *Traffic Assignment and Resource Allocation in Transportation Networks*. Baltimore, MD, USA: Johns Hopkins Univ., 1968.

[21] T. Leventhal, G. Nemhauser, and L. Trotter, Jr., "A column generation algorithm for optimal traffic assignment," *Transp. Sci.*, vol. 7, no. 2, pp. 168–176, 1973.

[22] R. Jayakrishnan, W. T. Tsai, J. N. Prashker, and S. Rajadhyaksha, "A faster path-based algorithm for traffic assignment," Univ. California Transp. Center, Berkeley, CA, USA, Tech. Rep. UCTC No. 191, 1994.

[23] A. Chen, Z. Zhou, and X. Xu, "A self-adaptive gradient projection algorithm for the nonadditive traffic equilibrium problem," *Comput. Oper. Res.*, vol. 39, no. 2, pp. 127–138, 2012.

[24] O. Perederieieva, M. Ehrgott, A. Raith, and J. Y. T. Wang, "A framework for and empirical study of algorithms for traffic assignment," *Comput. Oper. Res.*, vol. 54, pp. 90–107, Feb. 2015.

[25] Y. Nie, "A note on Bar-Gera's algorithm for the origin-based traffic assignment problem," *Transp. Sci.*, vol. 46, no. 1, pp. 27–38, 2012.

[26] T. Nakagaki, H. Yamada, and Á. Tóth, "Intelligence: Maze-solving by an amoeboid organism," *Nature*, vol. 407, no. 6803, p. 470, 2000.

[27] T. Nakagaki, H. Yamada, and M. Hara, "Smart network solutions in an amoeboid organism," *Biophys. Chem.*, vol. 107, no. 1, pp. 1–5, 2004.

[28] T. Nakagaki, R. Kobayashi, Y. Nishiura, and T. Ueda, "Obtaining multiple separate food sources: Behavioural intelligence in the physarum plasmodium," *Proc. Roy. Soc. London B Biol. Sci.*, vol. 271, no. 1554, pp. 2305–2310, 2004.

[29] A. Tero, R. Kobayashi, and T. Nakagaki, "A mathematical model for adaptive transport network in path finding by true slime mold," *J. Theor. Biol.*, vol. 244, no. 4, pp. 553–564, 2007.

[30] A. Tero *et al.*, "Rules for biologically inspired adaptive network design," *Science*, vol. 327, no. 5964, pp. 439–442, 2010.

[31] X. Zhang, A. Adamatzky, F. T. S. Chan, S. Mahadevan, and Y. Deng, "Physarum solver: A bio-inspired method for sustainable supply chain network design problem," *Ann. Oper. Res.*, doi: 10.1007/s10479-017-2410-x.

[32] X. Zhang *et al.*, "A biologically inspired network design model," *Sci. Rep.*, vol. 5, Jun. 2015, Art. no. 10794.

[33] D. Straszak and N. K. Vishnoi, "Natural algorithms for flow problems," in *Proc. 27th Annu. ACM SIAM Symp. Discrete Algorithms (SODA)*, Arlington, TX, USA, 2016, pp. 1868–1883.

[34] L. Liu, Y. Song, H. Zhang, H. Ma, and A. V. Vasilakos, "Physarum optimization: A biology-inspired algorithm for the Steiner tree problem in networks," *IEEE Trans. Comput.*, vol. 64, no. 3, pp. 819–832, Mar. 2015.

[35] M.-A. I. Tsompanas, G. C. Sirakoulis, and A. I. Adamatzky, "Evolving transport networks with cellular automata models inspired by slime mould," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1887–1899, Sep. 2015.

[36] X. Zhang *et al.*, "Solving 0-1 knapsack problems based on amoeboid organism algorithm," *Appl. Math. Comput.*, vol. 219, no. 19, pp. 9959–9970, 2013.

[37] A. Adamatzky, *Physarum Machines: Computers From Slime Mould*, vol. 74. Singapore: World Sci., 2010.

[38] Y. Liu *et al.*, "Solving NP-hard problems with Physarum-based ant colony system," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 14, no. 1, pp. 108–120, Jan./Feb. 2017, doi: 10.1109/TCBB.2015.246349.

[39] A. I. Adamatzky, "Route 20, autobahn 7, and slime mold: Approximating the longest roads in USA and Germany with slime mold on 3-d terrains," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 126–136, Jan. 2014.

[40] X. Zhang, Z. Zhang, Y. Zhang, D. Wei, and Y. Deng, "Route selection for emergency logistics management: A bio-inspired algorithm," *Safety Sci.*, vol. 54, pp. 87–91, Apr. 2013.

[41] V. V. Dixit and L. Denant-Boemont, "Is equilibrium in transport pure Nash, mixed or stochastic?" *Transp. Res. C Emerg. Technol.*, vol. 48, pp. 301–310, Nov. 2014.

[42] V. Bonifaci, K. Mehlhorn, and G. Varma, "Physarum can compute shortest paths," *J. Theor. Biol.*, vol. 309, pp. 121–133, Sep. 2012.

[43] V. Bonifaci, "Physarum can compute shortest paths: A short proof," *Inf. Process. Lett.*, vol. 113, nos. 1–2, pp. 4–7, 2013.

[44] T. Nakagaki *et al.*, "Minimum-risk path finding by an adaptive amoebal network," *Phys. Rev. Lett.*, vol. 99, no. 6, 2007, Art. no. 068104.

[45] A. Tero, R. Kobayashi, and T. Nakagaki, "Physarum solver: A biologically inspired method of road-network navigation," *Phys. A Stat. Mech. Appl.*, vol. 363, no. 1, pp. 115–119, 2006.

[46] P. Candito and G. D'aguì, "Three solutions for a discrete nonlinear Neumann problem involving the p-Laplacian," *Adv. Difference Equations*, vol. 2010, no. 1, pp. 1–11, 2010.

[47] N. K. Vishnoi, "Laplacian solvers and their algorithmic applications," *Theor. Comput. Sci.*, vol. 8, nos. 1–2, pp. 1–141, 2012.

[48] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proc. 36th Annu. ACM Symp. Theory Comput.*, Chicago, IL, USA, 2004, pp. 81–90.

[49] *Laplacian Solver Development*. Accessed on Mar 3, 2016. [Online]. Available: https://sites.google.com/a/yale.edu/laplacian/

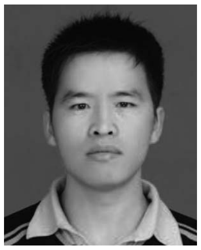
[50] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[51] C.-K. Park, S. Lee, and S. Park, "A label-setting algorithm for finding a quickest path," *Comput. Oper. Res.*, vol. 31, no. 14, pp. 2405–2418, 2004.

[52] Y. Sheffi, *Urban Transportation Network: Equilibrium Analysis With Mathematical Programming Methods*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1985.

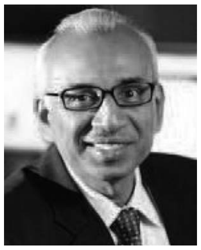
[53] *US Bureau of Public Roads, Traffic Assignment Manual*, U.S. Dept. Commerce, Washington, DC, USA, 1964.

[54] P. D. Site, F. Filippi, and C. Castaldi, "Reference-dependent stochastic user equilibrium with endogenous reference points," *Eur. J. Transp. Infrastruct. Res.*, vol. 13, no. 2, pp. 147–168, 2013.



Xiaoge Zhang received the B.S. degree from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2011, and the M.S. degree from Southwest University, Chongqing, in 2014. He is currently pursuing Ph.D. degree with Vanderbilt University, Nashville, TN, USA.

He has published over ten papers in international peer-reviewed journals, such as *Decision Support Systems*, *Annals of Operations Research*, *Reliability Engineering and System Safety*, and the *International Journal of Production Research*. His current research interests include uncertainty quantification, reliability assessment, risk analysis, network optimization, and evolutionary computation.



Sankaran Mahadevan received the B.S. degree from the Indian Institute of Technology, Kanpur, India, the M.S. degree from Rensselaer Polytechnic Institute, Troy, NY, USA, and the Ph.D. degree from the Georgia Institute of Technology, Atlanta, GA, USA.

He is a John R. Murray Sr. Professor of Engineering and a Professor of Civil and Environmental Engineering with Vanderbilt University, Nashville, TN, USA, where he has been serving since 1988. His research contributions are documented in over 500 publications, including two textbooks on reliability methods and 220 journal papers. He has directed 40 Ph.D. dissertations and 24 M.S. theses, and has taught several industry short courses on reliability and risk analysis methods. His current research interests include uncertainty quantification, model verification and validation, reliability and risk analysis, design optimization, system health monitoring, and integrated multiphysics simulation, with applications to civil, mechanical, and aerospace systems. His research has been extensively funded by National Science Foundation, National Aeronautics and Space Administration (NASA), Federal Aviation Administration, Department of Energy, Department of Defense, Department of Transportation, National Institute of Standards and Technology, General Electric, General Motors, Chrysler, Union Pacific, American Railroad Association, Sandia, Idaho, Los Alamos, and Oak Ridge National Laboratories.

Dr. Mahadevan was a recipient of the NASA Next Generation Design Tools Award, the SAE Distinguished Probabilistic Methods Educator Award, SEC Faculty Award, and best paper awards in the MORS Journal and the Structural Dynamics and Materials and International Modal Analysis Conference.