Routing Multiple Vehicles Cooperatively: Minimizing Road Network Breakdown Probability

Hongliang Guo, Zhiguang Cao, Madhavan Seshadri, Jie Zhang, Dusit Niyato, *Fellow, IEEE*, and Ulrich Fastenrath

***Abstract*—Traffic congestion has always been an impending challenge for drivers as well as traffic authorities. It causes frustrations to millions of passengers. The estimated financial cost is $2,200 billion per year in developed countries worldwide. In this paper, we propose an intelligent routing algorithm to minimize the traffic jam occurrence through directing the paths of multiple vehicles cooperatively. According to Kerner's breakdown minimization principle, we can claim that the traffic network optimum has been achieved if the probability for spontaneous traffic jam occurrence over the entire road network during a given observation time period is minimized. The proposed multivehicle routing approach is fully scalable and distributed, which essentially makes it directly applicable to real traffic networks such as that in Singapore. Through numerical studies, the proposed algorithm is much faster in terms of convergence speed than that of state-of-the-art distributed computation approaches. Moreover, our approach always maintains a feasible route guidance solution during the computation process, which is applicable to scenarios with real time decision making requirements, i.e., the reaction time must be within seconds. Simulation results in arbitrarily large road networks with realistic settings show the effectiveness of the proposed algorithm.**

***Index Terms*—Kerner's BM Principle, large scale network, multivehicle routing, matrix manipulation, Newton's method, traffic jam alleviation.**

I. Introduction

With the ever-increasing demands for mobility and modern logistics, the vehicle population has been steadily growing over the past several decades from 980 million units in 2009 to 1.015 billion units in 2010, reported by [1]. The adverse consequence of the vehicle population growth is traffic congestion. Nowadays, almost all the (metropolitan) cities such as Los Angeles, Beijing, and New York, are suffering from heavy traffic congestion. Published statistics from various sources reveal a staggering record: according to [2], Americans spend 14.5 million hours every day stuck in traffic, trying to commute or move goods to market. In [3, p. 10], the authors mentioned that traffic jams in the United States had wasted 5.5 billion hours of time, and 2.9 billion gallons of fuel that summed up to $121 billion in 2011. In [4], it is estimated that the costs of congestion in Britain, France, Germany, and the United States had amounted to $200 billion in 2013. This figure is expected to increase to $300 billion by 2030.

Alleviating traffic congestion benefits both traffic authorities and individual drivers. Therefore, different agencies have proposed and applied various operational approaches to reduce traffic congestion [5], [6]. The most prominent ones are a) limiting the total number of vehicles in the road network (forbidding $\frac{1}{5}$ of the total vehicle population to drive on the road based on the last digit of plate number in Beijing), b) incentivizing drivers to use congestion-free roads through Electronic Road Pricing (ERP) system such as in Singapore and London and c) promoting public transport usage (trains for long distance and bicycles for short distances for example).

In spite of all these traffic management approaches, traffic congestion is still a universal problem for both drivers and traffic engineers. Different from enforcing some public policies through government initiatives, e.g., a traffic management department, such as restricting certain vehicles from highways and charging money for 'normally-congested' areas, in this paper, we are trying to alleviate the traffic jam problem from the perspective of routing vehicles cooperatively.

In particular, this paper considers and adopts the three-phase traffic optimum theory for transportation networks proposed by Kerner in [7], [8] as the guidance for the overall traffic optimization. In Kerner's breakdown minimization principle, the network optimum can be achieved when the overall traffic jam occurrence probability is minimized. In view of this, the optimization problem of routing multiple vehicles can be formulated as minimizing the probability of traffic network breakdown occurrence while ensuring the actual arrival of all the individual drivers. Specifically, Kerner in [9] stated that the overall traffic network optimum could be achieved if the probability for spontaneous traffic breakdown occurrence over the entire road network is minimized for a given observation time period. In this paper, we apply a series of smart computational method to decompose the large-scale matrix manipulation problem into small ones, and make it executable on modern computers. As such, it can be deemed as computational intelligence. We will present the mathematical problem formulation with this objective in Section II-B.

Manuscript received October 12, 2016; revised January 2, 2017 and January 21, 2017; accepted January 26, 2017. Date of publication February 7, 2017; date of current version March 23, 2017. This work was supported by the BMW, Germany. Hongliang Guo and Zhiguang Cao contributed equally to this work. *(Corresponding Author: Zhiguang Cao.)*

H. Guo is with the School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu 610051, China (e-mail guohl1983@uestc.edu.cn).

Z. Cao is with the Energy Research Institute, Nanyang Technological University, Singapore 639798 (e-mail zhiguangcao@ntu.edu.sg).

M. Seshadri, J. Zhang, and D. Niyato are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail madhavan001@ntu.edu.sg; zhangj@ntu.edu.sg; dniyato@ntu.edu.sg).

U. Fastenrath is with the BMW Group, Munich D-80788, Germany (e-mail Ulrich.Fastenrath@bmw.de).

Digital Object Identifier 10.1109/TETCI.2017.2665592

A. Related Work

Routing multiple vehicles cooperatively has been gaining increasing interest from researchers over the past two decades. The literature in this research area can be roughly categorized into two groups. One is studying from the perspective of road network equilibrium while the other is investigating from the individual agent's perspective.

1) Network Equilibrium's Perspective: When drivers distribute themselves by their individual driving preferences such as minimizing their travel time, a certain user equilibrium [10] will be formed, in which no user can benefit more by making different decisions, e.g., selecting another route. Hence, the approaches from the network's perspective, usually apply a certain optimization or control methods, instantiated by certain public policies, over the vehicular traffic network. They aim at finding the equilibrium over the traffic network [9], [11]–[13] which assures network level optimum. In recent years, it was highlighted in [14] that the Wardrop's principles are usually not applicable for real dynamic traffic networks that are far from equilibrium and stationarity [12]. In particular, under the state of traffic jam (as has been shown by Wahle and Schreckenberg with colleagues [15] and Davis [14]), in most cases, no true Wardrop's equilibrium can be achieved.

Kerner pointed out that the adverse effect of traffic jam occurrence has been severely underestimated. Traffic jam usually leads to complex spatial-temporal congestion propagation, and thus will severely downgrade the overall traffic network's behaviour. Its impact to the involved drivers in terms of travel time, fuel consumption is hard to be predicted beforehand. Therefore, Kerner [9] proposed an innovative traffic network-level objective which aims at minimizing the overall traffic network breakdown occurrence. He argues that the optimum of a traffic network is reached when dynamic traffic optimization and/or control are performed in the network in such a way that the probability for spontaneous occurrence of traffic breakdown in at least one of the network bottlenecks during a given observation time is minimized [9]. This is equivalent to the maximization of the probability that traffic breakdown occurs at none of the network bottlenecks.

As we believe that the traffic behavior (travel speed/time) under jammed states is complex to model and predict, Kerner's traffic network optimization objective (minimizing road breakdown probability) is desirable for both traffic engineers and drivers. Thus, this paper adopts it as the optimization objective.

2) Agent's Perspective: Agents are autonomous software entities which try to achieve their objectives through interacting with the environment as well as other agents [16]. With their ability of autonomy and social interactions, agents are naturally used by many researchers for collaborative path planning of multiple vehicles [17]–[24].

Agents can represent vehicles, drivers, and/or other traffic elements, i.e., traffic lights. They are explicitly presented as active entities interacting with each other in the road network. Different agents may exhibit different information processing and decision making capabilities. Agents' behavior can be monitored, visualized and validated at the individual level, leading to new possibilities for debugging, analyzing and illustrating various traffic phenomena [25]. This paper roughly divides the multi-agent approaches within the transportation domain into 3 groups, namely, multi-agent vehicle coordination [17], [26]–[29], multi-agent traffic signal control [20], [30], [31], and multi-agent hybrid control [22], [32].

The multi-agent vehicle coordination approach [17], [26]–[29] models vehicles as agents. Naturally, multiple vehicles are represented by agents, and they will negotiate with each other to achieve better routing strategies. Vehicle agents can be reactive and perform rule-based behaviors [26] for path planning, or they can be proactive and learn [21], [23], [27], [28] from the environment and make decisions accordingly.

Another perspective of solving the multi-vehicle routing problem is to model traffic lights [20], [30], [31] as agents, and they will communicate and reach joint optimal traffic light schedules which can improve the overall traffic flow.

Recently, a new trend of hybrid agent-based control is emerging. This approach models both vehicles and traffic lights as agents [22], [32]. Jiang *et al.* [32] used a co-evolutionary approach to coordinate vehicles and traffic signals together so as to alleviate traffic jam situation. Promising results have been demonstrated in simulation.

Summary: In general, multi-agent algorithms are scalable with traffic networks as well as the number of involved computation entities. This is because the corresponding algorithm can be naturally distributed and support large number of agents with acceptable performance. However, these approaches can hardly guarantee a network-level optimum, i.e., road network breakdown minimization. By contrast, the network-based algorithms take the traffic network's global performance as the objective. In this case, the global objective is well defined and theoretically achievable. However, these algorithms usually cannot scale well with the number of participating vehicles and the size of road networks.

B. Contributions

In our paper, we aim at unifying the advantages of both the agent-based and network-based approaches through developing a fast scalable distributed approach. In particular, our algorithm can yield a recommended route to each cooperative vehicle. Additionally, the algorithm can guarantee a system-level road network breakdown minimization.

In the algorithm, we introduce a novel matrix decomposition technique during the process of network optimization. We guarantee that the equality constraint (which corresponds to the valid driving paths of all the vehicles) are always satisfied during the computation process, which ensures feasibility during computation. The algorithm utilizes the second order derivative information of the objective function. From optimization theory, when the objective function is self-concordant [33], which is the case for our traffic jam minimization problem, the second order derivative information provides a good guidance during the searching process for the optimum point. Thus, our proposed algorithm is better than traditional optimization approaches which utilize only the gradient information in terms of convergence speed.

Besides faster convergence speed than that of canonical distributed optimization algorithms, i.e., dual decomposition, our algorithm also provides an appealing feature, which is the guarantee of solution feasibility during the computation process. As we know, dual decomposition only reaches the feasible point in the final iteration, which means that the feasible solution can be only obtained in the end of the computation process, making it not applicable to scenarios with real time constraints.

The contributions of the paper can be summarized as follows:

1) the paper proposes a distributed approach to solve the system-level traffic optimization problem, achieving both network-level optimum and scalability;
2) the paper considers scenarios which contain default uncontrollable traffic in the road network;
3) the proposed distributed approach always ensures better convergence speed than that of state-of-the-art optimization techniques, which is shown in the Section VI;
4) the algorithm guarantees to maintain valid driving paths for all the vehicles during the update procedure, which cannot be achieved by other distributed optimization techniques.

C. *Organization of the Paper*

The following sections of the paper is organized as follows: Section II describes the fundamental traffic theory for our approach, and presents the mathematical problem formulation. We formally introduce our proposed distributed computation method in Section III, and then analyze both computation and communication complexity in Section IV. After considering two realistic aspects of the cooperative routing in Section V, we proceed with a theoretical simulation in Section VI to test the accuracy and speed of the algorithm compared with canonical algorithms. Section VII gives conclusion and potential future work.

II. Problem Formulation for Multivehicle Routing

A. *Kerner's Traffic Breakdown Minimization Principle*

Under small enough road network inflow rates, drivers move at their desired (or permitted) speeds. Usually, there are several alternative routes from an origin to a destination in a network for which the travel times are different but close to each other. When network inflow rates continuously increase, traffic jam occurs due to traffic breakdown, causing a sharp increase in the route travel time [9], [34]–[36]. Therefore, one of the theoretical problems in traffic networks is to find an optimal traffic assignment among alternative routes which prevents traffic breakdown under high enough network inflow rates while maintaining free flow states in the network.

From empirical (measured) traffic data, traffic breakdown at a specific urban road segment exhibits the following fundamental empirical characteristics [37], [38]:

1) The traffic breakdown probability is an increasing function of the flow rate, which can be well approximated through cumulative distribution function (CDF), and
2) A well-known hysteresis phenomenon associated with traffic breakdown is that when traffic breakdown has occurred at a road link, the outflow rate (throughput) of the road link is much smaller than its free flow rate.

The second empirical characteristic suggests us that when traffic breakdown occurs, the outflow rate is reduced significantly, which deteriorates the performance of the overall traffic network. Therefore, the objective should be to find the best way to avoid the traffic jam occurrence.

In this paper, we assume that the traffic breakdown model has already been given, and the probability of traffic breakdown occurrence is larger than zero (meaning that traffic breakdowns would occur), and our goal is to direct the traffic flow so that the overall traffic breakdown probability is minimized. In the following subsections, we will present the problem formulation and highlight the difficulties of solving the problem.

B. *Mathematical Problem Formulation*

According to Kerner's traffic flow theory [9], an optimum of a traffic network is achieved when dynamic traffic optimization and/or control are performed in the network in such a way that the probability for the occurrence of traffic breakdown in any of network links during a given observation time is minimized. This is equivalent to the maximization of the probability that traffic breakdown occurs at none of the network links.

Therefore, our objective is to maximize the probability that none of the network links encounters a traffic breakdown, which is stated as follows:

$$P = \prod_{i=1}^{n} \left(1 - \mathrm{Prob}(r_i + x_i)\right), \tag{1}$$

where r_i refers to the uncontrollable network load at the i's edge. $\mathrm{Prob}(\cdot)$ is the network breakdown function, and x_i is the load caused by the cooperative vehicles. n is the total number of road links in the traffic network. Here, the load of a road link refers to the expected vehicle density over that road link. Controllable load is the load that is caused by the cooperative routing vehicles, which can be directed to other road links. Default load refers to the load caused by other drivers which are out of control of the cooperative routing policy.

As we are routing multiple vehicle cooperatively, the minimal requirement is that we have to ensure that all the vehicles are able to reach their respective destinations. In mathematical expression, guaranteeing the arrival at destinations of all the vehicles can be expressed as the following network flow conservation equation:

$$\boldsymbol{W}\boldsymbol{x} = \boldsymbol{b} \tag{2}$$

where $\boldsymbol{W}$ is the node-incidence matrix describing the network topology, which is known beforehand, and $\boldsymbol{b}$ is describing the origin and destination of the vehicles. Note that the origin-destination descriptor (b) describes the macroscopic destination and origin of the vehicles rather than one vehicle. $\boldsymbol{W} \in \mathbb{R}^{m \times n}$, where m and n correspond to the number of nodes and the number of edges in the road network, respectively. Moreover, the controllable load $\boldsymbol{x}$ needs to be non-negative, as defined in

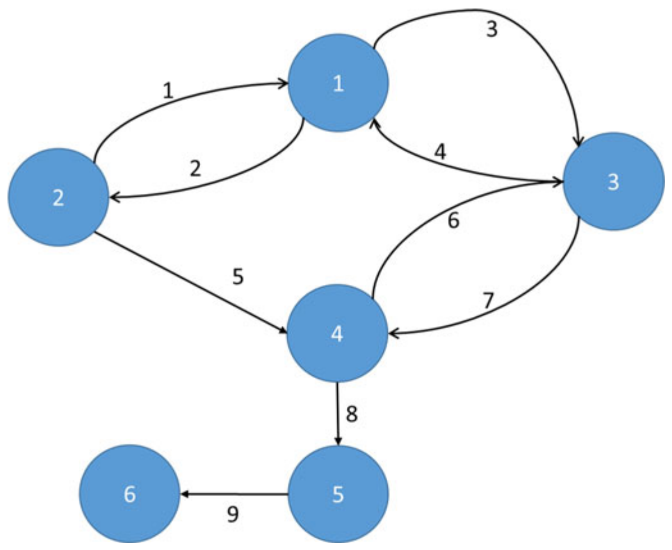

Fig. 1. Example road network.

the following equation:

$$\boldsymbol{x} \succeq \mathbf{0}. \tag{3}$$

Fig. 1 is an example in which there are six nodes and nine edges. In this simple example, $m = 6, n = 9$, the node incidence matrix $\boldsymbol{W}$ is represented as:

$$\mathbf{W} = \begin{pmatrix} -1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

.

Suppose a vehicle is traveling from node 1 to node 6, the origin-destination (OD) representation vector $\boldsymbol{b}$ is represented as $\mathbf{b} = \left(1, 0, 0, 0, 0, -1\right)^{\top}$. One of the routes can be encoded as $\boldsymbol{x} = (0, 1, 0, 0, 1, 0, 0, 1, 1)^{\top}$, which takes edges 2, 5, 8, and 9, sequentially.

As it has been emphasized in [9], the traffic breakdown probability of a given link can be well approximated by any cumulative distribution function (CDF). Thus, we take the following equation representing the network breakdown function, i.e.,

$$p(x) = \frac{\mathrm{e}^{wx+c}}{1 + \mathrm{e}^{wx+c}} \tag{4}$$

in which p represents the traffic breakdown occurrence probability, and x is the traffic load. w represents the unit impact of road traffic load to breakdown probability, and c represents other factors' influence, e.g., raining condition, on traffic breakdown. We assume that the r_i, w and c are already given beforehand. We emphasize here that (4) exhibits the CDF characteristic in which the value gradually increases to one when we increase the inflow rate and decreases to zero when we lower the inflow rate.

We transform the objective function in (1) by first applying logarithmic transformation of (1), changing the sign of the objective function to make the minimization problem and taking into account of (4), i.e.,

$$\begin{aligned} \min_{\boldsymbol{x}}.: \quad & -\ln\left(\prod\left(1 - \mathrm{Prob}(r_i + x_i)\right)\right) \\ &= \sum_{i=1}^{n} \ln\left(1 + \mathrm{e}^{w_i(x_i + r_i) + c_i}\right). \end{aligned} \tag{5}$$

Incorporating the objective function in (5) and the related constraints in (2) and (3), we can formulate our problem as follows:

$$\begin{aligned} \min_{\boldsymbol{x}}. \quad & \sum_{i=1}^{n} \ln\left(1 + \mathrm{e}^{w_i(x_i + r_i) + c_i}\right), \\ \text{s.t.} \quad & \boldsymbol{W}\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \succeq \mathbf{0}. \end{aligned} \tag{6}$$

The solution to the problem as given in (6) is defined as $\boldsymbol{d}^*$, which will be used as the input in the realistic aspect study in Section V.

III. Methodology

This section presents the logical flow of the algorithm, followed by the details of the proposed distributed algorithm. We apply the interior point method (IPM) [33] to solve the mathematical problem and propose a series of novel matrix manipulation algorithms to distribute the core large-scale matrix inversion process. The algorithm will find the optimal solution when IPM is finished. We are essentially proposing an efficient distributed IPM for the large scale convex optimization problem.

As the objective function in (6) is convex and all the constraints are affine, the problem is a convex optimization problem [33]. However, unlike canonical convex optimization problems, we are facing the large scale problem. Matrix $\boldsymbol{W}$, which corresponds to the road network representation, is large impeding us from directly using IPM [33].

A. Distributed Matrix Decomposition in Newton Step

We first directly apply IPM and then distribute the central heavy step into small steps. To ease the presentation, we omit the subscript of the objective function in (6) and define a generic function as follows:

$$g(x, r, w, c) = \ln\left(1 + \mathrm{e}^{w(x+r)+c}\right). \tag{7}$$

We start IPM [33] by initializing $t_0 = 1$ and setting $\mu_0 = 10$, which are canonical IPM-required parameters. In the k^{th} computation step, we are solving the problem (with barrier function) with $\mu = k\mu_0$ as described in (8):

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \mu t \sum_{i=1}^{n} f(x_i, r_i, w_i, c_i) - \sum_{i=1}^{n} \ln(x_i), \\ \text{subject to} \quad & \boldsymbol{W}\boldsymbol{x} = \boldsymbol{b}. \end{aligned} \tag{8}$$

The core computation step is to solve the following problem:

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{W}^{\top} \\ \boldsymbol{W} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{v} \end{pmatrix} = -\begin{pmatrix} \nabla f(\boldsymbol{x}) \\ \boldsymbol{W}\boldsymbol{x} - \boldsymbol{b} \end{pmatrix}, \tag{9}$$

where

$$\boldsymbol{H} = \mathrm{diag}\left(\mu t \nabla^2 f(x_i) + \frac{1}{x_i^2}\right) \tag{10}$$

where ∇ is the gradient operation.[1] Here, it is prohibitively complex to solve (9) directly when matrix $\boldsymbol{W}$ is very large. Therefore, it needs to be solved through elimination as:

$$\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top \Delta \boldsymbol{v} = \boldsymbol{W}\boldsymbol{x} - \boldsymbol{b} - \boldsymbol{W}\boldsymbol{H}^{-1} \bigtriangledown \boldsymbol{f}(\boldsymbol{x}). \tag{11}$$

We need to compute the inverse of $\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top$ in a distributed manner. After achieving the inverse of $\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top$, we are able to compute $\Delta \boldsymbol{w}$. $\Delta \boldsymbol{x}$ can then be obtained in the following equation:

$$\boldsymbol{H}\Delta\boldsymbol{x} = -(\nabla \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{A}^\top \Delta \boldsymbol{v}). \tag{12}$$

Note that $\boldsymbol{H}$ is a diagonal matrix, we only need to calculate the inverse of its diagonal elements. Therefore, it is trivial to compute the solution of (12) with given $\Delta\boldsymbol{v}$. In the following, we present an approach delivering the efficient solution of inversing $\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top$.

B. Matrix Decomposition for $\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top$ Inversion

The distributed solution to obtain the inverse of $\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top$ is presented in the following. The inputs to the distributed solution are $\boldsymbol{W} \in \boldsymbol{R}^{m\times n}$ and $\boldsymbol{H} \in (\mathrm{diag})^{n\times n} \cap \boldsymbol{S}^n_{++}$.

The algorithm works as follows:

1) we first decompose $\boldsymbol{W}$ along its column as:

$$\boldsymbol{W} = \big(\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_N\big) \tag{13}$$

in which N is denoted as the amount of computers, $\boldsymbol{W}_i \in \boldsymbol{R}^{m\times n_i}$; $\sum_{i=1}^N n_i = n$.

2) we compute $\boldsymbol{H}^{-\frac{1}{2}}$ and decompose it ($\boldsymbol{H}^{-\frac{1}{2}}$) along its row as in (14). Note that since $\boldsymbol{H} \in (\mathrm{diag})^{n\times n} \cap \boldsymbol{S}^n_{++}$, the calculation of $\boldsymbol{H}^{-1}$ is to inverse its diagonal elements. Define $\boldsymbol{H}^{-\frac{1}{2}}$ and we can get $\boldsymbol{H}^{-\frac{1}{2}}(\boldsymbol{H}^{-\frac{1}{2}})^\top = \boldsymbol{H}^{-1}$.
We calculate the square root of $\boldsymbol{H}^{-1}$'s diagonal element and obtain $\boldsymbol{H}^{-\frac{1}{2}}$. Then, we represent it as $\boldsymbol{\Lambda} = \boldsymbol{H}^{-\frac{1}{2}}$, which is in the form of:

$$\boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_1 \\ \boldsymbol{\Lambda}_2 \\ \vdots \\ \boldsymbol{\Lambda}_N \end{pmatrix} \tag{14}$$

where $\boldsymbol{\Lambda}_i \in \boldsymbol{R}^{n_i \times n}$.

3) We represent $\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top$ following the first two steps, and align (13) and (14). We can reach:

$$\begin{aligned}
&\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top \\
&= (\boldsymbol{W}_1, \ldots, \boldsymbol{W}_N) \begin{pmatrix} \boldsymbol{\Lambda}_1 \\ \boldsymbol{\Lambda}_2 \\ \vdots \\ \boldsymbol{\Lambda}_N \end{pmatrix} \begin{pmatrix} \boldsymbol{\Lambda}_1 \\ \boldsymbol{\Lambda}_2 \\ \vdots \\ \boldsymbol{\Lambda}_N \end{pmatrix}^\top (\boldsymbol{W}_1, \ldots, \boldsymbol{W}_N)^\top, \\
&= \left(\sum_{i=1}^N \boldsymbol{W}_i \boldsymbol{\Lambda}_i\right) \left(\sum_{i=1}^N \boldsymbol{W}_i \boldsymbol{\Lambda}_i\right)^\top, \\
&= \sum_{i=1}^N \sum_{j=1}^N \boldsymbol{W}_i \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_j^\top \boldsymbol{W}_j^\top.
\end{aligned} \tag{15}$$

$\boldsymbol{\Lambda}$ is diagonal, which gives an orthogonal decomposition basis in $\boldsymbol{R}^n$, i.e.,

$$\boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_j^\top = \begin{cases} \boldsymbol{0}, & \text{if } i \neq j, \\ \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top, & \text{if } i = j. \end{cases}$$

Therefore, (15) can be transformed in the following way:

$$\begin{aligned}
\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top &= \sum_{i=1}^N \textstyle\sum_{j=1}^N \boldsymbol{W}_i \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_j \boldsymbol{W}_j, \\
&= \sum_{i=1}^N \boldsymbol{W}_i \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top \boldsymbol{W}_i^\top.
\end{aligned} \tag{16}$$

Currently, another transformation of $\boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top$ in (16) is needed. Recall $\boldsymbol{\Lambda}$ in (14), since $\boldsymbol{\Lambda}$ is an $n \times n$ diagonal matrix, thus we can say that $\boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top$ is an $n_i \times n_i$ diagonal matrix. We then define

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top \tag{17}$$

and define $\boldsymbol{\Sigma}_i^{\frac{1}{2}} \in \boldsymbol{R}^{n_i \times n_i}$ as the square root matrix of $\boldsymbol{\Sigma}_i$. We get:

$$\boldsymbol{\Sigma}_i^{\frac{1}{2}} (\boldsymbol{\Sigma}_i^{\frac{1}{2}})^\top = \boldsymbol{\Sigma}_i. \tag{18}$$

Now, We transform (16) to:

$$\begin{aligned}
\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top &= \sum_{i=1}^N \boldsymbol{W}_i \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top \boldsymbol{W}_i^\top, \\
&= \sum_{i=1}^N \boldsymbol{W}_i \boldsymbol{\Sigma}_i^{\frac{1}{2}} (\boldsymbol{\Sigma}_i^{\frac{1}{2}})^\top \boldsymbol{W}_i^\top.
\end{aligned} \tag{19}$$

Here, we highlight that the computation of $\boldsymbol{W}_i \boldsymbol{\Sigma}_i^{\frac{1}{2}}$ is simple, because $\boldsymbol{\Sigma}_i^{\frac{1}{2}}$ is diagonal and we can obtain the result by executing the product over matrix $\boldsymbol{W}_i$ column-wisely. We can obtain:

$$\boldsymbol{W}_i \boldsymbol{\Sigma}_i^{\frac{1}{2}} = \boldsymbol{U}_i \boldsymbol{\Lambda}_i' \boldsymbol{V}_i^\top \tag{20}$$

from Singular Value Decomposition (SVD), in which $\boldsymbol{U}_i$, $\boldsymbol{V}_i$ are $m_i \times m_i$ unitary matrices, and $\boldsymbol{\Lambda}_i'$ is diagonal.

[1]Note that, throughout this paper, all gradient ∇ are with respect to x, e.g., $\nabla f(x_i) = \frac{\partial f}{\partial x}|_{x=x_i}$.

```
Algorithm 1: The computation process of the algorithm.
1  Initialization;
2  Input $\boldsymbol{W}$ and $\boldsymbol{H}$ to the algorithm;
3  Representation;
4  represent $\boldsymbol{W}$ as $(\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_N)$, represent
     $\boldsymbol{H}^{-1} = \boldsymbol{\Lambda} \times \boldsymbol{\Lambda}^\top$, partition $\boldsymbol{\Lambda}$ as $(\boldsymbol{\Lambda}_1^\top, \boldsymbol{\Lambda}_2^\top, \ldots, \boldsymbol{\Lambda}_N^\top)^\top$
     (please refer to (14));
5  computation steps;
6  (1) partition data into $N$ parts,
     $\boldsymbol{M} = \sum_{i=1}^N \boldsymbol{W}_i^\top \boldsymbol{\Lambda}_i^\top \boldsymbol{\Lambda}_i \boldsymbol{W}_i$;
7  (2) perform SVD over $\boldsymbol{\Lambda}_i \boldsymbol{W}_i$ respectively
     $\boldsymbol{\Lambda}_i \boldsymbol{W}_i = \boldsymbol{U}_i \boldsymbol{\Sigma}_i \boldsymbol{V}_i \Rightarrow \boldsymbol{M} = \sum_{i=1}^N \boldsymbol{V}_i^\top \boldsymbol{\Sigma}_i \boldsymbol{V}_i$;
8  (3) merge two matrices when they 'meet';
9  (3.1) compute $\boldsymbol{P}^{-1}$;
10 $\boldsymbol{U_q} = svd(\boldsymbol{\Lambda}_i^{\frac{1}{2}} \boldsymbol{V}_i^\top \boldsymbol{V}_j \boldsymbol{\Lambda}_j^{\frac{1}{2}})$;
11 $\boldsymbol{P}^{-1} = (\boldsymbol{V}_i \boldsymbol{\Lambda}_i^{\frac{1}{2}} \boldsymbol{U}_q)^\top$;
12 (3.2) calculate $\boldsymbol{M}_{ij}$;
13 $\boldsymbol{M_{ij}} = (\boldsymbol{P}^{-1})^\top (\boldsymbol{I} + \boldsymbol{\Lambda}_q \boldsymbol{\Lambda}_q^\top) \boldsymbol{P}^{-1}$;
14 (3.3) standardize $\boldsymbol{M}_{ij}$ (perform singular value
     decomposition over $\boldsymbol{M}_{ij}$;
15 $\boldsymbol{M}_{ij} = \boldsymbol{V}_{ij}^\top \boldsymbol{\Sigma}_{ij} \boldsymbol{V}_{ij}$;
16 (4) treat $\boldsymbol{M}_{ij}$ as a normal component;
17 if there is only one component in the system then
18     return $\boldsymbol{M}_{ij}$;
19 else
20     and go to (3) at Line 8 ;
```

Furthurmore, we transform (19) to:

$$\begin{aligned}\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top &= \sum_{i=1}^N \boldsymbol{W}_i \boldsymbol{\Sigma}_i^{\frac{1}{2}} (\boldsymbol{\Sigma}_i^{\frac{1}{2}})^\top \boldsymbol{W}_i^\top, \\ &= \sum_{i=1}^N \boldsymbol{U}_i \boldsymbol{\Lambda}_i' \boldsymbol{V}_i^\top \boldsymbol{V}_i (\boldsymbol{\Lambda}_i')^\top \boldsymbol{U}_i^\top, \\ &= \sum_{i=1}^N \boldsymbol{U}_i \boldsymbol{\Lambda}_i' (\boldsymbol{\Lambda}_i')^\top \boldsymbol{U}_i^\top. \end{aligned} \tag{21}$$

The term $\boldsymbol{\Lambda}_i'(\boldsymbol{\Lambda}_i')^\top$ is diagonal, and is defined as:

$$\boldsymbol{\Gamma_i} = \boldsymbol{\Lambda}_i'(\boldsymbol{\Lambda}_i')^\top. \tag{22}$$

Eq. (21) can now be expressed as follows:

$$\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top = \sum_{i=1}^N \boldsymbol{U}_i \boldsymbol{\Gamma}_i \boldsymbol{U}_i^\top. \tag{23}$$

Then, we calculate the inverse of Eq. 23's the right hand side (RHS) term, the detail of which is presented in Appendix I-A.

The whole computation process incurs a considerable amount of mathematical transformations (e.g., SVD) and matrix operations. As the key computation happens in computing the inverse of $\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top$, we outline, in Algorithm III-B, the pseudo code of the distributed computation process for obtaining an easy-to-inverse representation of $\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top$.

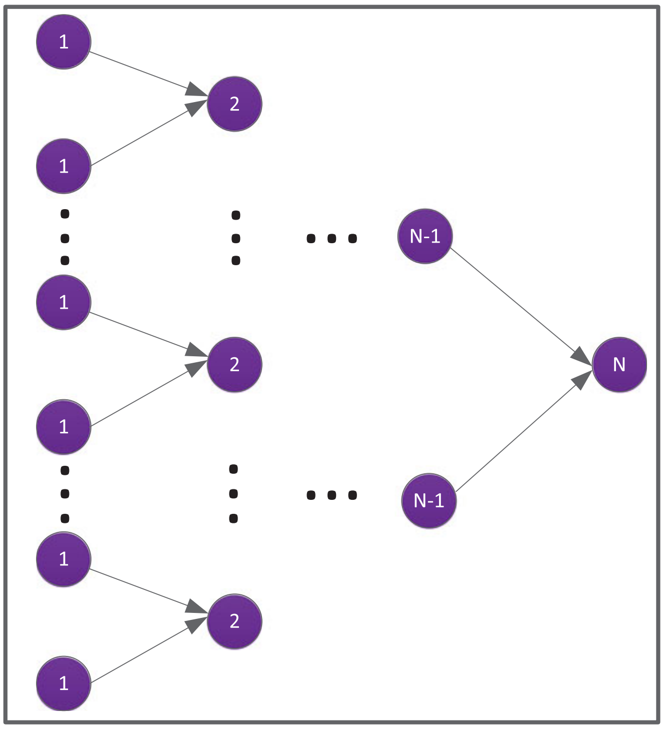

Fig. 2. Illustration of the matrix merging process.

Fig. 2 illustrates the matrix merging process of the proposed algorithm. Each computation entity begins with a local description of the map data ($\boldsymbol{W}_i \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top \boldsymbol{W}_i^\top$). After N iterations of matrix merging, we have a final representation form of the big matrix $\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top$.

C. Practical Use Case

Now we have finished the logical flow process of the proposed methodology, the real work flow of the cooperative routing engine will be as follows:

1) The users of the cooperative routing service input their final destination as in a typical GPS navigation system.
2) The back-end cooperative routing engine collects all the origin-destination pairs from the vehicles participated in the cooperative routing service. Based on the current road traffic condition, we compute the expected traffic load of all the road segments by solving (6), which is the core idea proposed in this paper
3) After solving (25), the final driving parameter, i.e., the driving path which is denoted as $\boldsymbol{x}$, will be returned to the users.

IV. Analysis

A. Computational Complexity

This subsection presents the complexity analysis of the proposed algorithm. We first present the metric gauging computation complexity.

1) Baseline Computational Complexity (Flops): In modern computers, the number of floating-point operations (flops) serves as a good indicator of the computation cost. A flop is defined as the basic operation between two floating-point numbers, such as addition, subtraction, multiplication or division [33]. In order to evaluate the algorithm's computational complexity, we can count the required flops, and represent it as a function of the problem's dimensions and then keeping only the dominating terms.

According to the flop definitions, the computational complexity of the SVD operation over a matrix $\boldsymbol{W} \in \boldsymbol{R}^{m\times n}$ can be expressed as $O(m^2n + n^3)$. The computational complexity of $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$, in which $\boldsymbol{A} \in \boldsymbol{R}^{m\times n}$ and $\boldsymbol{B} \in \boldsymbol{R}^{n\times p}$, can be expressed as $O(mnp)$.

2) Worst Case Computational Complexity Evaluation: This paper proposes a distributed approach, therefore, computing the overall computational complexity may not be valid. Instead, we should focus on how the computational effort is distributed among the computers. In the following, we compute the computational complexity over one single computation entity. Note that the partition process may not be absolutely equal, moreover the computational process over one computer may exit before the whole approach exits. Therefore, we only consider the the 'worst' entity's computational complexity. Note that, the 'worst computation entity' refers to the computation entity performing the most complex computation over the whole process.

Evaluating the computation process of the approach in Appendix I-A (Matrix Merging Process) and Algorithm III-B, we need to apply the SVD over $\boldsymbol{X}_i$, which requires $O(m_i^2n + n^3)$ flops. Because we consider the 'worst computational entity', we express the computational complexity as $O(\max_i\,(m_i^2)n + n^3)$.

After that, the approach calculates $\boldsymbol{U}_q$ and $\boldsymbol{P}^{-1}$. $\boldsymbol{U}_q$ needs three operations of $n\times n$ matrix-matrix product, and one $n\times n$ matrix singular value decomposition . Note that both matrix product's and SVD's computational complexity are $O(n^3)$. Therefore, computing $\boldsymbol{U}_q$'s computational complexity is $O(n^3)$. Similarly, the computational complexity of obtaining $\boldsymbol{P}^{-1}$ is $O(n^3)$. Multiplying those two matrices requires $O(n^3)$ flops. The complexity of performing SVD over the final matrix is $O(n^3)$. Unifying the terms and dropping constant ones, we reach that the computational complexity is $O(n^3)$.

The computational complexity analysis in the previous paragraph is only for one step of matrix merging. Now we need to calculate the number of times of matrix merging that we actually need. Note that, altogether, there are N computational entities. After one epoch of matrix merging, we have $\lceil \frac{N}{2} \rceil$ computation entities, where $\lceil x \rceil$ is a ceiling operator which returns the smallest integer larger than x. Thus, after at most $\lceil \log_2 N \rceil$ times of matrix merging, the algorithm finishes.

Therefore, the computational complexity for the worst computation entity is $O(\max_i\,(m_i^2)n + n^3 + \lceil \log_2 N \rceil n^3)$. Dropping relevant constant terms, we reach $O(\max_i\,(m_i^2)n + \lceil \log_2 N \rceil n^3)$.

B. Communication Cost

The scalability of the distributed algorithm, with respect to the number of computation entities, can be gauged by the total communication cost over all the computation entities. Thus, in this paper, we perform the communication cost analysis by analyzing how the total communication cost scales with the number of computation entities. First, we need to quantify the communication cost.

The cost of one communication event is usually related to the used bandwidth, latency, and waiting time spent [39]. The cost of one communication event in our proposed algorithm can be summarized into a single abstract cost, denoted as γ. Here, we emphasize that for the high-level communication cost analysis, it is not uncommon to assume a single abstract cost for one communication event [40].

The total communication cost (ϑ) of the proposed algorithm can be expressed as follows:

$$\vartheta = \gamma N_{\text{com}} \tag{24}$$

where N_{com} is the total number of communication events.

In the proposed algorithm, there is no back-and-forth communication between computation entities. Moreover, once there is communication between two computation entities, one computation entity is merged and leaves the system. It means that when there are N computation entities in the network, there are exactly $N-1$ communication events to finish the computation. Thus, the total communication cost as expressed in (24) is $\gamma(N-1)$.

To this end, we can conclude that the communication cost of our algorithm is $O(N)$, indicating that the communication cost of the algorithm scales linearly with the number of computation entities.

V. Two Other Realistic Aspects

A. Path Computation for Multiple Vehicles

The output of the cooperative routing engine, i.e., the solution to the problem as defined in (6), is not the route for every vehicle. Instead, the output is the extra flow rates to be allocated to the cooperative vehicles. Thus, we still need a routing function to compute a viable route for each of the cooperative vehicles. Suppose that the output of the cooperative routing engine is $\boldsymbol{d}^*$, we can define the path planning problem as follows:

$$\begin{aligned}
\underset{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_N}{\text{minimize}} \quad & \sum_{i=1}^{N} \boldsymbol{c}^\top \boldsymbol{x}, \\
\text{subject to} \quad & \boldsymbol{M}\boldsymbol{x} = \boldsymbol{b}, \\
& \boldsymbol{x} = \{0,1\}^{Nn}, \\
& \sum_{i=1}^{N} \boldsymbol{x}_i \preceq \boldsymbol{d}^* + \boldsymbol{1}
\end{aligned} \tag{25}$$

where N is the total number of cooperative vehicles

$$\boldsymbol{c} = \left(\boldsymbol{c}_1^\top, \boldsymbol{c}_2^\top, \ldots \boldsymbol{c}_N^\top\right)^\top, \tag{26}$$

$$\boldsymbol{b} = \left(\boldsymbol{b}_1^\top, \boldsymbol{b}_2^\top, \ldots, \boldsymbol{b}_N^\top\right)^\top, \tag{27}$$

$$M = \begin{pmatrix} \boldsymbol{W} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{W} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \cdots & \mathbf{0} & \boldsymbol{W} \end{pmatrix} \tag{28}$$

and $\boldsymbol{x}$ denotes the driving paths of the N vehicles.

The solution of the problem given in (25) is the corresponding driving paths for all the cooperative routing vehicles. Here, we highlight that, in this paper, we do not focus on proposing an algorithm/mechanism to solve the problem given in (25). By contrast, our focus is to propose an efficient and distributed algorithm to solve the extra flow rate allocation problem. In order to efficiently solve the problem as defined in (25), we proposed a partial Lagrangian multiplier method in [6].

B. Leaving and Joining Vehicles

Since we are routing multiple vehicles cooperatively, there usually is the case where some vehicles are dynamically joining the 'cooperative routing club'[2], and some vehicles are leaving the club. We might be experiencing dynamically changing of origins and destinations. We next explain the routing policy adaptation.

In the proposed algorithm, we are essentially using IPM to solve the convex optimization problem. We present in Section III that in every iteration, we will use a backtrack method to find an appropriate value of α. When the origin-destination pairs are changing, i.e., $\boldsymbol{b}$ changes, what the cooperative routing needs to do is to omit the backtrack method for the first iteration and perform a 'full' Newton step (meaning to set $\alpha = 1$).

The proof can be given as follows:

A full Newton step refers to the step without line search.

Objective: verify

$$\boldsymbol{W}(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{b} \tag{29}$$

based on (11) and (12). Note that we do not have $\boldsymbol{W}\boldsymbol{x} = \boldsymbol{b}$ anymore.

Proof:

$$\begin{aligned} &\boldsymbol{W}(\boldsymbol{x} + \Delta\boldsymbol{x}) - \boldsymbol{b} \\ &= \boldsymbol{W}\boldsymbol{x} - \boldsymbol{b} - \boldsymbol{W}\boldsymbol{H}^{-1}(\triangledown f(\boldsymbol{x}) + \boldsymbol{W}^\top \boldsymbol{v}), \\ &= \boldsymbol{W}\boldsymbol{x} - \boldsymbol{b} - \boldsymbol{W}\boldsymbol{H}^{-1} \triangledown f(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top \boldsymbol{v}, \\ &= \boldsymbol{W}\boldsymbol{x} - \boldsymbol{b} - \boldsymbol{W}\boldsymbol{H}^{-1} \triangledown f(\boldsymbol{x}) \\ &\quad - (\boldsymbol{W}\boldsymbol{x} - \boldsymbol{b} - \boldsymbol{W}\boldsymbol{H}^{-1} \triangledown f(\boldsymbol{x})), \\ &= 0. \end{aligned} \tag{30}$$

Hence, once we have finished one full Newton step of the update, the cooperative routing is able to return the valid flow allocation to the road network. Here, the term 'valid' means that the traffic flow rate allocation is able to drive the cooperative routing vehicles to their respective destinations.

[2]Here, a cooperative routing club refers to a set of vehicles that are willing to share their origins and destinations and adhere to the directed driving path.

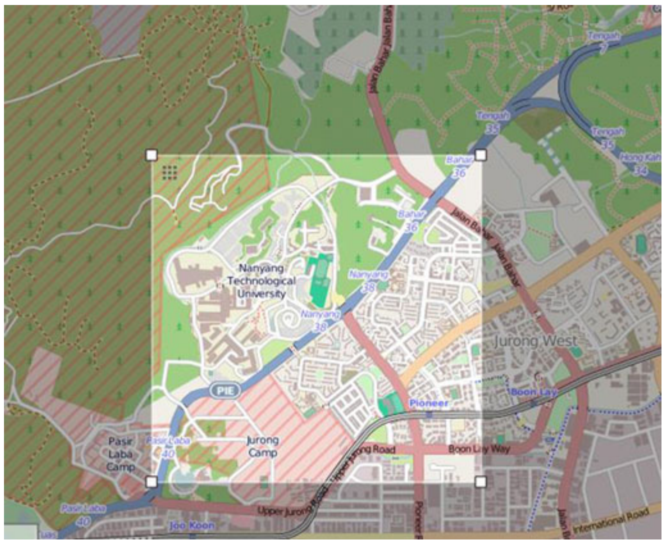

Fig. 3. The selected simulation environment.

VI. Simulation Results and Analysis

This section evaluates the performance of our proposed approach through answering the following questions:

1) Are we calculating $\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top$'s inverse correctly?
2) Can the proposed algorithm achieve a much faster convergence rate than state-of-the-art methods?
3) Can the approach guarantee the valid driving paths during the computation process?

A. Simulation Setup

1) Simulation Environment: In this section, we select a small area within Singapore as the environment, which is shown in Fig. 3. The area contains 1,703 nodes and 3,136 edges, which means that the matrix $\boldsymbol{W}$ in the equality constraint is of size 1,703 × 3,136. Note that we intentionally select a small road network as the environment. This is to ensure a feasible centralized solution to the optimization problem, which can be used as a benchmark to evaluate the accuracy of the proposed algorithm.

2) Parameter Setup: The parameters in (4) are set as follows: $c = -3, w = 0.01$. The total number of vehicles in the simulation environment is $100/w = 10,000$. r_i is set to be uniformly distributed between 0% and 90% of the total loads. This setting ensures that at least 10% of the total vehicles can be controlled in the simulation environment.

IPM-related parameters are set as follows: $\epsilon = 10^{-4}$, $t = 1$, and $u = 10$. The related parameters inside Newton's method are set as $\alpha = 0.3$ and $\beta = 0.8$, respectively, and the Newton decrements stopping criterion is set as: $\epsilon_{nt} = 10^{-4}$. The selection of the parameters are according to the Newton parameter selection strategy in convex optimization theory [33].

3) Simulation Results and Analysis: Fig. 4 shows the 'error' between our distributed algorithm and the centralized algorithm.

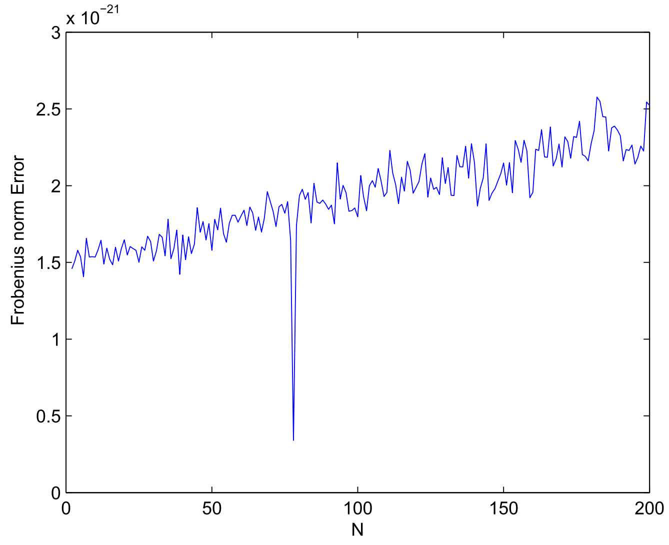

Fig. 4. Computation error against the number of computation entities.

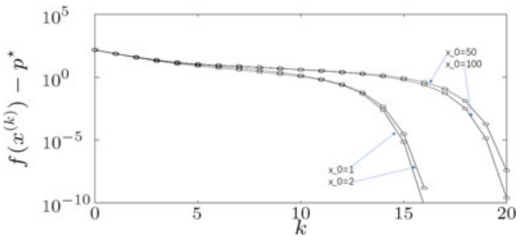

Fig. 5. The difference between the k^{th}-step solution and the optimum (obtained through the centralized method). Note that different curves are for different initial starting points.

Here, we calculate the inverse of $\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top$ through a centralized method and our proposed distributed algorithm. Then, the error is defined as the Frobenius norm [41] over the difference between two resulting matrices. In modern digital computers (as error always exists), when the error is in the scale of 10^{-10}, it can be deemed as accurate. The Frobenius norm of a matrix is calculated as follows:

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} \tag{31}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$.

Fig. 4 shows the Frobenius error between the proposed algorithm and centralized solution. In the figure, although there is a slight increase when we increase N (the number of computation entities), it is safe to say that our algorithm reaches the correct inverse $\boldsymbol{W}\boldsymbol{H}^{-1}\boldsymbol{W}^\top$ because the errors are in the scale of 10^{-21}. Note that we do notice the sharp drop in the Frobenius norm error in Fig. 4 for N at around 78. We postulate that the underlying cause for the drop should lie in the inherent round up mechanism of the computer used in the computation.

Fig. 5 shows our algorithm's performance of convergence. Note that we compute the optimal solution through the IPM directly. And we implement our proposed distributed algorithm and compare the result with the pre-computed solution. We can see in Fig. 5, after around 20 Newton steps, the algorithm reaches solution with sufficiently small deviation (the error magnitude in the scale of 10^{-5}). In the experiment, we have performed the optimization process for 4 different initialization point, each of which corresponds to each curve in Fig. 5, and all of them converge after about 20 Newton steps.

Here, it is worth noting that, the number of Newton steps (20 in the case of our simulation) will stay more or less the same as the size of the problem grows. This phenomenon implies that the number of required Newton steps would still be around 20 for a large network. Nonetheless, the theoretical bound of Newton steps grows polynomially (i.e., O($\sqrt{m}$)) with the size of the problem. However, computational complexity for every Newton step increases exponentially with the size of the network. Thus, our main focus in the algorithm is to make the Newton step computation distributed and hence scalable.

In the simulation, we have also used dual decomposition and ADMM [42] to get the distributed solution and the number of iterations for the convergence of both methods is around *7 million*, which is reasonable as dual decomposition techniques and ADMM only guarantee convergence in *infinite* number of iterations. However, we cannot simply conclude that dual decomposition method is slower than our proposed algorithm, because each computation step within dual decomposition is much simpler than that of a Newton step computation. To establish a fair complexity comparison, we need to evaluate the computation effort spent over each step for dual decomposition and our proposed algorithm.

The computational complexity (in terms of the number of flops as stated in Section IV) of each iteration in ADMM or dual decomposition is $O(mn)$, where m and n are the number of nodes and edges, respectively. The computational complexity in each iteration of our algorithm is $O(mn_i^2)$, where n_i is the number of edges allocated to the computation entity. In the extreme case ($n_i = n$), the computational complexity in each iteration of our algorithm is $n = 3,703$ times of that in dual decomposition. Multiplying the total required number of steps together, and defining the computational complexity of each iteration in dual decomposition as a unit, we can say that the total computational complexity of our algorithm is $21 \times 3,703 = 77,763$ flops. While the total computational complexity of dual decomposition and ADMM is *7 million* (meaning that they require 7×10^6 flops [33] of operation). Apparently, our algorithm is significantly better in terms of computational complexity.

Fig. 6 shows the feasibility metric during the optimization process. We wish to note here that maintaining feasibility during the optimization process is very important for our problem, as drivers would not like to wait very long for the final suggested route, returning a near-optimal route in real time is desirable. Whether a point at the k^{th} iteration is feasible in mathematical optimization depends on whether the point satisfies all the constraints.

Since we already use IPM [33] with log barrier function, it is guaranteed that the inequality constraint is always satisfied (or else, the term $\ln(x)$ will not be valid), and thus, during the validation process, we only verify whether the equality constraint

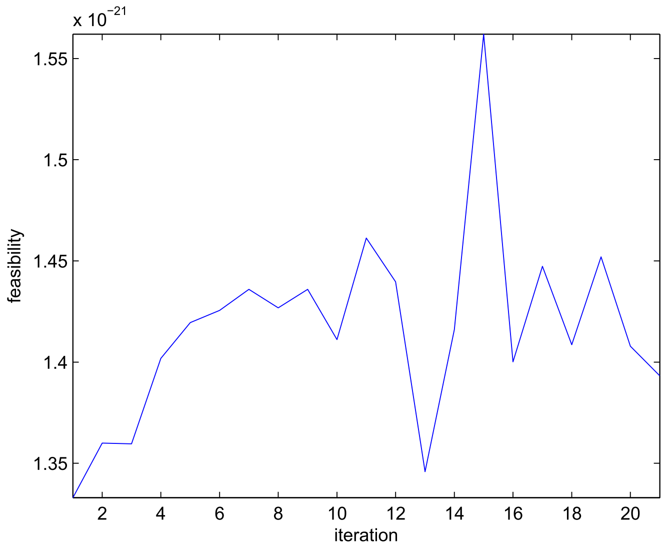

Fig. 6. Feasibility evaluation (Here, the feasibility metric is defined as the Euclidean norm of $\boldsymbol{W}\boldsymbol{x} - \boldsymbol{b}$).

holds. Here, we wish to note that both ADMM and dual decomposition merge the equality constraint into the objective (i.e., forming the Lagrangian objective function) and then perform corresponding decomposition operations. In this case, they can only achieve a feasible point after the whole optimization process is finished. As shown in Fig. 6, for every Newton step, the feasibility metric is always in the scale of 10^{-21}, which means that it is always feasible.

4) Discussions: So far, we have demonstrated that our proposed algorithm can return accurate numerical results in large scale traffic networks. Furthermore, during the optimization process, the algorithm always guarantees a feasible solution as shown in Fig. 6. This property is particularly important when the cooperative router is required to return the routing decision within a restricted time frame (say, a few seconds). We are testing our algorithm in simulators with Kerner-Klenov microscopic stochastic model to further verify the applicability of the algorithm, and will report related results in the near future.

VII. Conclusion and Future Works

We have displayed a distributed optimization approach to cooperatively route multiple vehicles under Kerner's BM principle. Different from state-of-the-art techniques, we make the Newton-step calculation process distributed through a novel matrix decomposition algorithm. The novel algorithm enables us to capitalize on the second derivative information, which eventually speeds up the optimization process. Moreover, a feasible solution can be guaranteed during the iteration process. Simulation results have demonstrated both aforementioned advantages. In the future, we will test our algorithm in simulators with Kerner-Klenov microscopic stochastic model to further verify the applicability of the algorithm.

Currently, we assume that the traffic network breakdown probability is known. In the future, we would like to employ different methods to obtain the traffic network breakdown probability model. Additionally, we would also like to apply the proposed algorithm to the real usage and conduct experiments. Theoretically, if at least 10% of the vehicles are following our routing strategy, we can significantly alleviate the overall traffic jam occurrence probability as shown in the simulation. Our vendor (BMW) has provided us with open access to all car-sharing vehicles in Munich (accounting for 8% of the total vehicle population in Munich). We are on track of implementing the cooperative router into the real BMW navigation system and testing the algorithm's effectiveness in real environment.

Appendix

A. Matrix Merging Process

We aim to solve the inverse of $\sum \boldsymbol{V}_i \boldsymbol{\Lambda}_i \boldsymbol{V}_i^\top$ in a distributed manner. The term can be represented as:

$$\boldsymbol{M} = \sum_{i=1}^{N} \boldsymbol{V}_i \boldsymbol{\Lambda}_i \boldsymbol{V}_i^\top \tag{32}$$

in which Λ_i is diagonal matrix and $\boldsymbol{V}_i$ is unitary matrix. Ideally, if $\boldsymbol{M}$ can be represented in the following form:

$$\boldsymbol{M} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top \tag{33}$$

in which $\boldsymbol{V}$ is a Unitary Matrix, and $\boldsymbol{\Lambda}$ is diagonal, whose diagonal values are positive, then calculating the inverse of $\boldsymbol{M}$ is an easy task.

Before trying to transform (32) into the form of (33) in one shot, we begin by combining two matrices which is much easier. Now, we need to transform $\boldsymbol{V}_i \boldsymbol{\Lambda}_i \boldsymbol{V}_i^\top + \boldsymbol{V}_j \boldsymbol{\Lambda}_j \boldsymbol{V}_j^\top$ into (33). We define

$$\boldsymbol{M}_{ij} = \boldsymbol{V}_i \boldsymbol{\Lambda}_i \boldsymbol{V}_i^\top + \boldsymbol{V}_j \boldsymbol{\Lambda}_j \boldsymbol{V}_j^\top. \tag{34}$$

As both terms are symmetric positive definite, based on Theorem VII.1 in the following subsection, we know that there is a non-singular matrix $\boldsymbol{P}$, satisfying $\boldsymbol{P}^\top \boldsymbol{V}_i \boldsymbol{\Lambda}_i \boldsymbol{V}_i^\top \boldsymbol{P} = \boldsymbol{I}$, and $\boldsymbol{P}^\top \boldsymbol{V}_j \boldsymbol{\Lambda}_j \boldsymbol{V}_j^\top \boldsymbol{P} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, where $\boldsymbol{I}$ refers to an identity matrix

Thus, matrix $\boldsymbol{P}$ has the property:

$$\boldsymbol{P}^\top \boldsymbol{M}_{ij} \boldsymbol{P} = \boldsymbol{I} + \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n). \tag{35}$$

After simple deduction, we can obtain

$$\begin{aligned} \boldsymbol{M}_{ij} &= (\boldsymbol{P}^{-1})^\top (\boldsymbol{I} + \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n))\boldsymbol{P}^{-1}, \\ &= (\boldsymbol{P}^{-1})^\top \text{diag}(\lambda_1 + 1, \lambda_2 + 1, \ldots, \lambda_n + 1)\boldsymbol{P}^{-1}. \end{aligned} \tag{36}$$

Continuing to apply SVD over the last part of (36), we transform the representation of $\boldsymbol{M}_{ij}$ as:

$$\boldsymbol{M}_{ij} = \boldsymbol{V}_{ij} \boldsymbol{\Lambda}_{ij} \boldsymbol{V}_{ij}^\top. \tag{37}$$

Now, $\boldsymbol{M}_{ij}$ is already represented in the same form as pre-merging. We are able to continue with the combining process when $\boldsymbol{M}_{ij}$ 'meets' another matrix. Thus, the most important computation step is how to calculate matrix $\boldsymbol{P}^{-1}$.

1) The Calculation Steps of $\boldsymbol{P}^{-1}$: Recall (34), since $\boldsymbol{\Lambda}_i$ is a positive diagonal matrix, assume $\boldsymbol{\Lambda}_i = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$, we define:

$$\boldsymbol{\Lambda}_i^{-\frac{1}{2}} = \mathrm{diag}\left(\lambda_1^{-\frac{1}{2}}, \lambda_2^{-\frac{1}{2}}, \ldots, \lambda_n^{-\frac{1}{2}}\right) \tag{38}$$

and then we define

$$\boldsymbol{P}_1 = \boldsymbol{\Lambda}_i^{-\frac{1}{2}} \boldsymbol{V}_i^\top. \tag{39}$$

Now, we have:

$$\boldsymbol{P}_1^{-1} = \boldsymbol{V}_i \boldsymbol{\Lambda}_i^{\frac{1}{2}}. \tag{40}$$

We multiply each term on the right hand side of (34) by $\boldsymbol{P}_1$ and by $\boldsymbol{P}_1^\top$. Then, obtain

$$\begin{aligned}\boldsymbol{P}_1 \boldsymbol{M}_{ij} \boldsymbol{P}_1^\top &= \boldsymbol{P}_1 \boldsymbol{V}_i \boldsymbol{\Lambda}_i \boldsymbol{V}_i^\top \boldsymbol{P}_1^\top + \boldsymbol{P}_1 \boldsymbol{V}_j \boldsymbol{\Lambda}_j \boldsymbol{V}_j^\top \boldsymbol{P}_1^\top, \\ &= \boldsymbol{I} + \boldsymbol{\Lambda}_i^{-\frac{1}{2}} \boldsymbol{V}_i^\top \boldsymbol{V}_j \boldsymbol{\Lambda}_j \boldsymbol{V}_j^\top \boldsymbol{V}_i (\boldsymbol{\Lambda}_i^{-\frac{1}{2}})^\top.\end{aligned} \tag{41}$$

Here, we define

$$\boldsymbol{\Lambda}_j^{\frac{1}{2}} = \mathrm{diag}(\lambda_1^{\frac{1}{2}}, \lambda_2^{\frac{1}{2}}, \ldots, \lambda_n^{\frac{1}{2}}). \tag{42}$$

Then, we can transform (41) to

$$\boldsymbol{P}_1 \boldsymbol{M}_{ij} \boldsymbol{P}_1^\top = \boldsymbol{I} + \boldsymbol{\Lambda}_i^{-\frac{1}{2}} \boldsymbol{V}_i^\top \boldsymbol{V}_j \boldsymbol{\Lambda}_j^{\frac{1}{2}} (\boldsymbol{\Lambda}_j^{\frac{1}{2}})^\top \boldsymbol{V}_j^\top \boldsymbol{V}_i (\boldsymbol{\Lambda}_i^{-\frac{1}{2}})^\top. \tag{43}$$

We then define

$$\boldsymbol{Q} = \boldsymbol{\Lambda}_i^{-\frac{1}{2}} \boldsymbol{V}_i^\top \boldsymbol{V}_j \boldsymbol{\Lambda}_j^{\frac{1}{2}}. \tag{44}$$

Eq. (43) is transformed to:

$$\boldsymbol{P}_1 \boldsymbol{M}_{ij} \boldsymbol{P}_1^\top = \boldsymbol{I} + \boldsymbol{Q}\boldsymbol{Q}^\top. \tag{45}$$

Applying the SVD on $\boldsymbol{Q}$, we can obtain

$$\boldsymbol{Q} = \boldsymbol{U}_q \boldsymbol{\Lambda}_q \boldsymbol{V}_q^\top \tag{46}$$

where $\boldsymbol{V}_q$ and $\boldsymbol{U}_q$ are Unitary Matrices, and $\boldsymbol{\Lambda}_q$ is diagonal. Applying (45) to (46), we have

$$\begin{aligned}\boldsymbol{P}_1 \boldsymbol{M}_{ij} \boldsymbol{P}_1^\top &= \boldsymbol{I} + \boldsymbol{U}_q \boldsymbol{\Lambda}_q \boldsymbol{V}_q^\top (\boldsymbol{U}_q \boldsymbol{\Lambda}_q \boldsymbol{V}_q^\top)^\top, \\ &= \boldsymbol{I} + \boldsymbol{U}_q \boldsymbol{\Lambda}_q \boldsymbol{V}_q^\top \boldsymbol{V}_q (\boldsymbol{\Lambda}_q)^\top \boldsymbol{U}_q^\top, \\ &= \boldsymbol{I} + \boldsymbol{U}_q \boldsymbol{\Lambda}_q (\boldsymbol{\Lambda}_q)^\top \boldsymbol{U}_q^\top, \\ &= \boldsymbol{U}_q \boldsymbol{U}_q^\top + \boldsymbol{U}_q \boldsymbol{\Lambda}_q (\boldsymbol{\Lambda}_q)^\top \boldsymbol{U}_q^\top, \\ &= \boldsymbol{U}_q (\boldsymbol{I} + \boldsymbol{\Lambda}_q \boldsymbol{\Lambda}_q^\top) \boldsymbol{U}_q^\top.\end{aligned} \tag{47}$$

Continue to solve (47), we get

$$\begin{aligned}\boldsymbol{M}_{ij} &= \boldsymbol{P}_1^{-1} \boldsymbol{U}_q (\boldsymbol{I} + \boldsymbol{\Lambda}_q \boldsymbol{\Lambda}_q^\top) \boldsymbol{U}_q^\top (\boldsymbol{P}_1^{-1})^\top, \\ &= \boldsymbol{V}_i \boldsymbol{\Lambda}_i^{\frac{1}{2}} \boldsymbol{U}_q (\boldsymbol{I} + \boldsymbol{\Lambda}_q \boldsymbol{\Lambda}_q^\top) \boldsymbol{U}_q^\top (\boldsymbol{V}_i \boldsymbol{\Lambda}_i^{\frac{1}{2}})^\top.\end{aligned} \tag{48}$$

Recall (35), P^{-1} can be directly computed as follows:

$$\boldsymbol{P}^{-1} = (\boldsymbol{V}_i \boldsymbol{\Lambda}_i^{\frac{1}{2}} \boldsymbol{U}_q)^\top. \tag{49}$$

2) Standardize the Merged Matrix: At present, we get $\boldsymbol{P}^{-1}$, and then $\boldsymbol{M}_{ij}$ is represented as follows:

$$\boldsymbol{M}_{ij} = (\boldsymbol{P}^{-1})^\top (\boldsymbol{I} + \boldsymbol{\Lambda}_q \boldsymbol{\Lambda}_q^\top) \boldsymbol{P}^{-1}. \tag{50}$$

$\boldsymbol{M}_{ij}$ is an $n \times n$ matrix, we apply SVD over $\boldsymbol{M}_{ij}$, $\boldsymbol{M}_{ij}$ can be represented as follows:

$$\boldsymbol{M}_{ij} = \boldsymbol{V}_{ij} \boldsymbol{\Lambda}_{ij} \boldsymbol{V}_{ij}^\top. \tag{51}$$

Now, we have shown that we can combine two matrices, who are represented in (33), and continue to transform the merged matrix into the standard form in (33). As it goes on, we can obtain the final matrix which is also in Eq. 33's form.

B. Theorem: Simultaneous Congruentization and Diagonalization Over Two Matrices

Theorem VII.1. $\forall\ \boldsymbol{A} \in \boldsymbol{S}_{++}^n$, $\boldsymbol{B} \in \boldsymbol{S}^n$, $\exists \boldsymbol{P}$, such that: (1) $\boldsymbol{P}^\top \boldsymbol{A} \boldsymbol{P} = \boldsymbol{I}$; (2) $\boldsymbol{P}^\top \boldsymbol{B} \boldsymbol{P}$ =diag$(\lambda_1, \lambda_2, \ldots, \lambda_n)$;(3) $\exists \boldsymbol{P}^{-1}$.

Proof. We have $\boldsymbol{A} \in \boldsymbol{S}_{++}^n$, thus $\boldsymbol{A}$ is congruent to the identity matrix, which means $\exists \boldsymbol{P}_1$, s.t. $\boldsymbol{P}_1^\top \boldsymbol{A} \boldsymbol{P}_1 = \boldsymbol{I}$.

$\boldsymbol{B} \in \boldsymbol{S}^n$, thus $\boldsymbol{P}_1^\top \boldsymbol{B} \boldsymbol{P}_1$ is also symmetric, then we are able to orthogonally make it a diagonal matrix. $\boldsymbol{B} \in \boldsymbol{S} \Rightarrow \boldsymbol{P}_1^\top \boldsymbol{B} \boldsymbol{P}_1 \in \boldsymbol{S} \Rightarrow \exists \boldsymbol{P}_2$ s.t. (1) $\boldsymbol{P}_2^\top \boldsymbol{P}_2 = \boldsymbol{P}_2 \boldsymbol{P}_2^\top = \boldsymbol{I}$; (2) $\boldsymbol{P}_2^\top \boldsymbol{P}_1^\top \boldsymbol{B} \boldsymbol{P}_1 \boldsymbol{P}_2 =$ diag$(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

Defining $\boldsymbol{P} = \boldsymbol{P}_1 \boldsymbol{P}_2$, obtain: $\boldsymbol{P}^\top \boldsymbol{A} \boldsymbol{P} = \boldsymbol{I}$, and in the meanwhile $\boldsymbol{P}^\top \boldsymbol{B} \boldsymbol{P}$=diag$(\lambda_1, \lambda_2, \ldots, \lambda_n)$. ■

Acknowledgment

The authors would like to thank L. Jian-Kai for implementing the algorithm on the SUMO simulator, and BMW for financial support.

References

[1] Wards Auto, "World vehicle population tops 1 billion units," Aug. 2011.
[2] ABC News, "The costs of highway congestion," Feb. 2001.
[3] Texas A&M Transportation Institute, "TTI's 2012 urban mobility report," 2012.
[4] The Economist, "The cost of traffic jams," Nov. 2014.
[5] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Finding the shortest path in stochastic vehicle routing: A cardinality minimization approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 6, pp. 1688–1702, Jun. 2016.
[6] Z. Cao, H. Guo, J. Zhang, and D. Niyato, "Improving the efficiency of stochastic vehicle routing: A partial lagrange multiplier method," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3993–4005, Jun. 2016.
[7] B. S. Kerner, "The maximization of the network throughput ensuring free flow conditions in traffic and transportation networks: Breakdown minimization (bm) principle versus wardrop equilibria," *Eur. Phys. J. B*, vol. 89, no. 9, pp. 89–199, 2016.
[8] B. Kerner, "Breakdown minimization principle versus wardrop equilibria for dynamic traffic assignment and control in traffic and transportation networks: A critical mini-review," *Physica A Statist. Mech. Appl.*, vol. 466, pp. 626–662, 2017.
[9] B. S. Kerner, "Optimum principle for a vehicular traffic network: Minimum probability of congestion," *J. Phys. A Math. Theoretical*, vol. 44, no. 9, pp. 41–66, 2011.
[10] J. Wardrop, *Wardrop of Some Theoretical Aspects of Road Traffic Research-Road Paper*. Road Engineering Division, 1952.
[11] Z. Cao, H. Guo, J. Zhang, F. Oliehoek, and U. Fastenrath, "Maximizing the probability of arriving on time: A practical q-learning method," in *Proc. 31th AAAI Conf. Artif. Intell.*, 2017.
[12] H. Rakha and A. Tawfik, "Traffic networks: Dynamic traffic routing, assignment, and assessment," in *Encyclopedia of Complexity and Systems Science*, R. A. Meyers, Ed. New York, NY, USA: Springer, 2009, pp. 9429–9470.
[13] N. Gartner and C. Stamatiadis, "Traffic networks, optimization and control of urban," in *Encyclopedia of Complexity and Systems Science*, R. A. Meyers, Ed. New York, NY, USA: Springer, 2009, pp. 9470–9500.
[14] L. Davis, "Realizing wardrop equilibria with real-time traffic information," *Physica A Statist. Mech. Appl.*, vol. 388, no. 20, pp. 4459–4474, 2009.
[15] J. Wahle, A. L. Bazzan, F. Klügl, and M. Schreckenberg, "Decision dynamics in a traffic scenario," *Physica A Statist. Mech. Appl.*, vol. 287, no. 3/4, pp. 669–681, 2000.

[16] L. Padgham and M. Winikoff, *Developing Intelligent Agent Systems: A Practical Guide*, vol. 13. New York, NY, USA: Wiley, 2005.
[17] J. Auld and A. Mohammadian, "Activity planning processes in the agent-based dynamic activity planning and travel scheduling (ADAPTS) model," *Transp. Res. Part A Policy Practice*, vol. 46, no. 8, pp. 1386–1403, 2012.
[18] M. Balmer, N. Cetin, K. Nagel, and B. Raney, "Towards truly agent-based traffic and mobility simulations," in *Proc. 3rd Int. Joint Conf. Auton. Agents Multiagent Syst. Volume 1*, 2004, pp. 60–67.
[19] Z. Cao, S. Jiang, J. Zhang, and H. Guo, "A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion," *IEEE Trans. Intell. Transp. Syst.*, vol. PP, no. 99, pp. 1–16, 2016.
[20] A. Bazzan, "A distributed approach for coordination of traffic signal agents," *Auton. Agents Multi-Agent Syst.*, vol. 10, no. 1, pp. 131–164, 2005.
[21] J. Pan, I. Popa, K. Zeitouni, and C. Borcea, "Proactive vehicular traffic rerouting for lower travel time," *IEEE Trans. Veh. Technol.*, vol. 62, no. 8, pp. 3551–3568, Oct. 2013.
[22] A. Bazzan, D. de Oliveira, F. Klügl, and K. Nagel, "To adapt or not to adapt consequences of adapting driver and traffic light agents," in *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, K. Tuyls, A. Nowe, Z. Guessoum, and D. Kudenko, Eds. Berlin, Germany: Springer, 2008, pp. 1–14.
[23] K. Tumer, A. K. Agogino, and Z. Welch, "Traffic congestion management as a learning agent coordination problem," in *Multiagent Architectures for Traffic and Transportation Engineering*, A. Bazzan and F. Kluegl, Eds. New York, NY, USA: Springer, 2009, pp. 261–279.
[24] Z. Cao, H. Guo, J. Zhang, and U. Fastenrath, "Multiagent-based route guidance for increasing the chance of arrival on time," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 3814–3820.
[25] A. Bazzan and F. Klügl, "A review on agent-based technology for traffic and transportation," *Knowl. Eng. Rev.*, vol. 29, pp. 375–403, 2014.
[26] A. Bazzan, J. Wahle, and F. Klügl, "Agents in traffic modelling from reactive to social behaviour," in *KI-99: Advances in Artificial Intelligence*, vol. 1701, W. Burgard, A. Cremers, and T. Cristaller, Eds. Berlin, Germany: Springer, 1999, pp. 303–306.
[27] E. Camponogara and J. Kraus, Werner, "Distributed learning agents in urban traffic control," in *Progress in Artificial Intelligence*, vol. 2902, F. Pires and S. Abreu, Eds. Berlin, Germany: Springer, 2003, pp. 324–335.
[28] C. Desjardins, J. Laumnier, and B. Chaib-draa, "Learning agents for collaborative driving," in *Multi-Agent Systems for Traffic and Transportation*, F. Bazzan, A. L. C. Klugl, Ed. Hershey, PA, USA: IGI Global, 2004, pp. 240–260.
[29] H. Dia, "An agent-based approach to modelling driver route choice behaviour under the influence of real-time information," *Transp. Res. Part C Emerg. Technol.*, vol. 10, no. 56, pp. 331–349, 2002.
[30] L. B. de Oliveira and E. Camponogara, "Multi-agent model predictive control of signaling split in urban traffic networks," *Transp. Res. Part C Emerg. Technol.*, vol. 18, no. 1, pp. 120–139, 2010.
[31] N. Gartner and N. R. C. U. T. R. Board, *OPAC: A Demand-Responsive Strategy for Traffic Signal Control*. Transportation Research Board, National Research Council, 1983.
[32] S. Jiang, J. Zhang, and Y.-S. Ong, "A pheromone-based traffic management model for vehicle re-routing and traffic light control," in *Proc. 2014 Int. Conf. Auton. Agents Multi-Agent Syst.*, 2014, pp. 1479–1480.
[33] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[34] B. S. Kerner, "Criticism of generally accepted fundamentals and methodologies of traffic and transportation theory: A brief review," *Physica A Statist. Mech. Appl.*, vol. 392, no. 21, pp. 5261–5282, 2013.
[35] B. Kerner, "Failure of classical traffic flow theories: Stochastic highway capacity and automatic driving," *Physica A Statist. Mech. Appl.*, vol. 450, pp. 700–747, 2016.
[36] B. S. Kerner and S. L. Klenov, "Phase transitions in traffic flow on multilane roads," *Phys. Rev. E*, vol. 80, no. 5, 2009, Art. no. 056101.
[37] B. S. Kerner, *Introduction to Modern Traffic Flow Theory and Control: The Long Road to Three-Phase Traffic Theory*. New York, NY, USA: Springer, 2009.
[38] B. S. Kerner, S. L. Klenov, and M. Schreckenberg, "Probabilistic physical characteristics of phase transitions at highway bottlenecks: incommensurability of three-phase and two-phase traffic-flow theories," *Phys. Rev. E*, vol. 89, no. 5, 2014, Art. no. 052807.
[39] G. Ballard, "Communication optimal parallel multiplication of sparse random matrices," EECS Dept., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2013-13, 2013.
[40] E. Klavins, "Communication complexity of multi-robot systems," J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, Eds. Berlin, Germany: Springer, 2003, pp. 275–292.
[41] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1986.
[42] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

Hongliang Guo received the Master's degree from Beijing Institute of Technology, Beijing, China, and the Ph.D. degree in the field of computer engineering from Stevens Institute of Technology, Hoboken, NJ, USA, in 2011. His research interest include multiagent technologies with applications to multirobot systems as well as multivehicle environment. He became a Program Leader of BMW future mobility lab in Nanyang Technological University, in 2013. In 2016, he joins the University of Electronic Science and Technology, Chenghua, China as an Associate Professor.

Zhiguang Cao received the Ph.D degree from Nanyang Technological University, Singapore in 2017. He graduated from Guangdong University of Technology, Guangzhou, China as a bachelor in 2009 and received his master degree from Nanyang Technological University, Singapore, in 2012. He is currently a Research Fellow with the BMWNTU Future Mobility Research Lab and Energy Research Institute @ NTU (ERIAN), Singapore. He is interested in the general topics of traffic and transportation optimization.

Madhavan Seshadri is currently working toward the Bachelor degree in computer science at Nanyang Technological University, Singapore. His Bachelor's thesis is associated with the Future Mobility Research Lab, Singapore on the topic of cooperative vehicle routing. He has prior internship experiences at BMW Group, Germany (in 2016) and Agency for Science Technology and Research, Singapore (in 2015). His research interests include cooperative vehicle routing, multiagent systems and machine learning.

Jie Zhang received the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2009. He also received the Alumni Gold Medal at the Convocation Ceremony in the same year, which is awarded to honor the top Ph.D. graduates from the University of Waterloo. He is currently an Associate Professor at the School of Computer Engineering, Nanyang Technological University, Singapore. He His papers appeared in many top journals as well as conferences, where he received several Best Paper Awards. He is an active member in several research communities.

Dusit Niyato (F'16) received the Ph.D. degree from the University of Manitoba, Winnipeg, MB, Canada, in 2008. He is currently an Associate Professor in the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include the areas of wireless communication, optimization, smart grid systems, and mobile cloud computing.

Ulrich Fastenrath is with BMW, Germany from May 2012. He is responsible for the research and development in traffic and routing domain, as well as its spin-off products, where he has extensive experience. Before moving to BMW, he managed the European traffic development team of Navteqs DDG Organization to deploy its real-time traffic capabilities across Europe. As a Project Manager in DDG from 1995, he later became the Head of Development to Spearhead DDGs Traffic Technology. Prior to his tenure at DDG, he was a System Developer at BOSCH, where he created the technology to run the main computer of BOSCHs automatic fee collection system.