Hindawi
Journal of Advanced Transportation
Volume 2023, Article ID 8862759, 15 pages
https://doi.org/10.1155/2023/8862759

Research Article

A Class of Efficient Algorithms for the Bi-Level Demand Adjustment Problems in Congested Traffic Networks

Lan Cheng,[1] **Jun Xie**,[1] **Jun Huang**,[1] **Liyang Feng**,[1] **Qianni Wang**,[2] **and Hongtai Yang**[1]

[1]*School of Transportation and Logistics, Southwest Jiaotong University, Chengdu, Sichuan 610031, China*
[2]*Department of Civil and Environmental Engineering, Northwestern University, Evanston, Illinois 60208, USA*

Correspondence should be addressed to Jun Huang; jun.huang@my.swjtu.edu.cn

Received 18 August 2023; Revised 12 October 2023; Accepted 14 November 2023; Published 9 December 2023

Academic Editor: Sheng Jin

This paper studies a class of gradient-descent heuristic algorithms for the bi-level demand adjustment problem (DAP), which seeks to adjust origin-destination (OD) matrices based on observed link flows in congested transportation networks. We first present a general gradient-descent solution framework for the bi-level DAP and then examine and further develop its two building blocks, namely, the gradient approximation and stepsize calculation. This paper presents two gradient approximation and four stepsize calculation methods, of which two stepsize methods are newly developed. Similarities and differences between these algorithms, as well as the relevant implementation issues are discussed in great detail. The numerical results show that algorithms employing the new stepsize calculation strategies consistently outperform existing algorithms in terms of both computational precision and efficiency.

1. Introduction

OD matrices (ODM) are essential inputs for many transportation models and applications, ranging from transportation planning to environmental impact assessment [1]. Despite the significance of ODM, accurately and efficiently estimating the travel demand between different OD pairs within a transportation network remains a persistent challenge in the field of transportation. Early methods mainly rely on large-scale sampled surveys like home interviews and roadside surveys, which are often criticized for being time-consuming, costly, and likely to produce biased and out-of-date results [2]. Given that the traffic count data on some selected links are much easier to obtain, a class of DAP that aims to estimate ODM from observed link flows has been extensively studied by many researchers since the pioneering work of Low [3]. The reader is referred to previous research [4, 5] for a comprehensive review of DAP.

DAP can be viewed as the reverse process of the traffic assignment problem because it takes observed link flows as input and seeks to determine the most probable ODM that would reproduce the link flows observed when assigned to the network [4]. In general networks; however, the information provided by observed link flows is insufficient to reliably infer a unique ODM because there are typically much more OD pairs than links. It should be noted that while the number of links is typically proportional to the node number, the number of OD pairs is typically proportional to the square of the node number. DAP thus requires some other information in addition to observed link flows, typically in the form of past data like historical ODM, which may come from earlier surveys or trip distribution models [6]. DAP models can therefore be broadly defined as a bi-level optimization problem [7], where the upper-level problem seeks to minimize the deviation between the adjusted ODM and the target matrix as well as the deviation between the assigned link flows and the observed link flows, and the lower-level problem is a traffic assignment problem that implicitly defines the mapping from ODM to link flows.

Depending on which type of route-choice assumption is employed in the lower-level problem, a variety of DAP models have been proposed in the literature, such as the proportion assignment models [8–12] and the user equilibrium models [13–16]. In the first class of models, fixed link or route choice proportions are typically used to model the relationship between ODM and link flows. This independence assumption between route choice behavior and demand levels has been widely acknowledged to be impractical when congestion effects are considered in travel. In contrast, the second class models the mapping between ODM and link flow pattern as the user equilibrium traffic assignment problem, in which the route choice proportions may vary in response to the adjustment of ODM during the estimation process. Since the optimal solutions of the user equilibrium traffic assignment problem can be reformulated as nonlinear complementary conditions, DAP with the user equilibrium assumption can also be formulated as mathematical programming with equilibrium constraints (MPEC), which presents several essential difficulties for the solution of DAP: First, it is a NP-hard problem [17] with poor mathematical properties, such as nonconvexity and nondifferentiability. Second, the mapping relationship between ODM and equilibrium flow is implicitly defined by the lower-level problem, rather than being explicitly known [18]. Third, the dimension of the actual ODM is extremely large, which poses a huge challenge for solving it efficiently [18].

Given the bi-level nature of DAP problems, most existing implementable algorithms for DAP are heuristic gradient methods, and the differences among them are primarily determined by how the gradient of the objective function is approximated. Spiess [13] designed a heuristic algorithm based on gradient descent by assuming that the path choice proportion is a local constant and calculating the approximate gradient of the objective function with respect to the OD demand. Yang [19] proposed the bi-level programming model for DAP based on sensitivity analysis and calculated the approximate value of the upper-level objective function derivative using the sensitivity analysis method proposed by Tobin and Friesz [20], which iteratively solves the upper- and lower-level problems. Chen and Florian [21] reformulated the DAP problem as a single-level optimization problem using general bi-level programming theory, where the marginal function of the lower-level user equilibrium problem is explicitly used as a constraint and the gradient function of the implicit marginal function is derived. Lundgren and Peterson [16] proposed a projection gradient method for adjusting ODM by solving a quadratic programming problem to obtain an approximate gradient value, which can be applied to solving medium and large networks. Walpen et al. [18] improved the algorithm proposed by Lundgren and Peterson [16] and proposed a descent scheme based on the gradient approximation of the DAP objective function.

As with any other gradient technique for mathematical programs, the convergence performance of gradient algorithms for DAP is affected not only by the quality of the descent direction but also by the method used to determine the line search stepsize. It should be noted that the choice of stepsize calculation approaches can have a significant effect on the overall convergence precision and efficiency of DAP algorithms because: (1) DAP is not differentiable at some points, making the choice of stepsize even more crucial. If the stepsize is too large, the algorithm may fail to converge and overshoot the minimum. If the stepsize is too tiny, the technique may converge slowly, which is computationally expensive and unsuitable for large-scale tasks. (2) The majority of line search methods for identifying a stepsize typically entail many evaluations of the objective function [22], and each evaluation necessitates the computation of a traffic assignment problem, which can be computationally costly for large traffic networks. Consequently, the efficiency and accuracy of the solving procedure are highly dependent on the stepsize approaches employed. However, few studies have undertaken research on the DAP stepsize determination approach.

This paper presents an algorithmic framework that generalizes the vast majority of previously presented approximate gradient algorithms for DAP. Within this framework, different gradient approximation methods and stepsize methods can be fairly compared to reveal their similarities and differences and more significantly, to provide insight into the observed computational performance that can be used to design more efficient DAP algorithms. Combining several gradient approximation methods and stepsize calculation methods, the paper also contributes several new algorithms to the family of DAP algorithms. First, it introduces new strategies for computing the stepsize, which, despite being essential to the algorithm's effectiveness, have not been extensively explored in the literature. Second, a number of new heuristics designed to solve the problem in large-scale networks are proposed for DAP. Our numerical experiments corroborate that the selection of stepsize methods employed in the search process has a significant effect on the performance of the algorithm.

The remainder of the paper is organized as follows: Section 2 introduces notations, formulations, and the algorithm framework for DAP. Section 3 presents the techniques for estimating gradient. Section 4 suggests new approaches to stepsize calculation. Section 5 demonstrates the effectiveness of the new algorithm using a comprehensive set of numerical experiments. The last section concludes the paper with a summary of the main findings and potential directions for future research.

2. Formulations and Algorithm Overview

Let us define $G = (N, A, I)$ as a congested traffic network, where N is the set of nodes and A is the set of links, and I denotes the set of OD pairs. Define $\mathbf{d} = \{d_i\}, i \in I$ as the OD demand matrix; $\widehat{\mathbf{d}} = \{\widehat{d}_i\}, i \in I$ is the historical OD demand matrix; $\mathbf{x} = \{x_a\}, a \in A$ is the traffic flow vector; $\widehat{\mathbf{x}} = \{\widehat{x}_a\}, a \in \widehat{A}$ is the observed traffic flow vector; $t_a, a \in A$ is the cost of link a; $h_k, k \in K_i, i \in I$ is the flow of the path k, where K_i denotes the set of paths used by OD pair i. δ_{ak} represents the relationship between a link and a path, if the path k uses link a, the value is 1; otherwise, it is 0. This paper formulates DAP as a bi-level programming problem, employing the generalized least squares (GLS) method. The GLS method offers the advantage of allowing a weighted combination of the historical OD demand matrix and observed link flow, considering the relative accuracy of these data sources [23]. This method has been widely adopted by many researchers [16, 18, 24, 25]. A detailed discussion can be found in Cascetta [26]. Thus, the bi-level programming formulations of DAP can be expressed as follows:

$$(U)\min Z(\mathbf{d}) = \theta_1 \cdot (\mathbf{d}-\widehat{\mathbf{d}})^T \cdot (\mathbf{d}-\widehat{\mathbf{d}}) + \theta_2 \cdot (\mathbf{x}(\mathbf{d})-\widehat{\mathbf{x}})^T \cdot (\mathbf{x}(\mathbf{d})-\widehat{\mathbf{x}}), \tag{1a}$$

$$\text{s.t.}\mathbf{d} \geq \mathbf{0}, \tag{1b}$$

where the link flow $\mathbf{x}$ is the optimal solution of the lower user equilibrium assignment model formulated as follows:

$$(L)\min f(\mathbf{x}) = \sum_{a\in A}\int_0^{x_a} t_a(w)\mathrm{d}w, \tag{1c}$$

$$\text{s.t.} \sum_{k\in K_i} h_k = d_i, i \in I, \tag{1d}$$

$$\sum_{i\in I}\sum_{k\in K_i}\delta_{ak}h_k = x_a, a \in A, \tag{1e}$$

$$h_k \geq 0, k \in K_i, i \in I, \tag{1f}$$

where θ_1 and θ_2 denote, respectively, the relative confidence level of the information contained about the historical OD demand and the observed traffic flow.

The objective function (1a) is a least squares function that aims to reduce the variance between the adjusted OD demand matrix and the historical OD demand matrix, as well as the error between the link flow generated by the adjusted OD demand and the observed link flow. It is worth noting that the significance of matching the adjusted OD demand with the historical OD demand and the adjusted link flow with the observed link flow is measured by the coefficients θ_1 and θ_2, respectively. It is reasonable to expect θ_1 and θ_2 to be close to 1 and 0, respectively, if the historical OD demand is determined from a large sample of high-quality data and the counting devices used to observe link flow are outdated and prone to errors. On the other hand, the θ_1 close to 0 and the θ_2 close to 1 are more appropriate if the veracity of historical OD demand is low and the observed link flow data's veracity is high. In practice, the values of θ_1 and θ_2 can be modified based on the specific situation.

The lower-level problem (1c)–(1f) is a user equilibrium assignment model that represents the mapping process from OD demand $\mathbf{d}$ to the equilibrium link flow $\mathbf{x}$, by minimizing the Beckmann function (1c) in order to fulfill the user equilibrium principle. As a constraint of the bi-level programming model, the lower-level problem itself is an equilibrium optimization problem. Because the feasible domain is typically nonconvex, finding the global optimal solution is generally difficult. As a result, heuristic algorithms and local search strategies are used in this research to find the local optimal solution in the initial subdomain. The acquisition of the optimal solution depends on the selection of the initial solution chosen.

We next present a coherent algorithmic framework that can accommodate various gradient approximation and stepsize calculation methods. This framework will first estimate the gradient of the objective function at the current point and project it into the feasible region to obtain a descent direction, and then determine an appropriate stepsize to move along the feasible descent direction to obtain a sufficient decrease in the objective function value. This procedure is carried out until the convergence criterion is met. The algorithm framework may be described in Algorithm 1.

It is noteworthy that the algorithm can accurately solve the gradient of the objective function and construct the descent direction when the objective function is differentiable at the current point. However, when the objective function is nondifferentiable at certain points, the gradient of the objective function may not exist, and partial derivatives can be used to express the objective function's descent direction. The algorithm presented in this paper optimizes the objective function by approximating its gradient and selecting an appropriate stepsize, which is applicable for real-world applications because exact solutions for gradients and stepsizes are theoretically and computationally demanding and are not conducive to solving large networks.

3. Approximation of the Gradient

In order to construct the descent direction of the upper-level model, it is necessary to determine the rate of change in link flow as a result of OD demand fluctuation during the process of solving DAP. Conceptually, the link flow $\mathbf{x}$ can be defined as a function of OD demand $\mathbf{d}$, i.e., $x_a(\mathbf{d}), a \in A$. Then, the objective function of the upper-level model can be reformulated as follows:

$$\min Z(\mathbf{d}) = \theta_1 \cdot Z_1(\mathbf{d}) + \theta_2 \cdot Z_2(\mathbf{x}(\mathbf{d})). \tag{2a}$$

Consequently, the gradient of $Z(\mathbf{d})$ can be expressed as follows:

$$\nabla Z(\mathbf{d}) = \theta_1 \cdot \nabla_{\mathbf{d}} Z_1(\mathbf{d}) + \theta_2 \cdot \nabla_{\mathbf{d}} \mathbf{x}(\mathbf{d}) \cdot \nabla_{\mathbf{x}} Z_2(\mathbf{x}(\mathbf{d})), \tag{2b}$$

$$\nabla_d Z_1(\mathbf{d}) = \left\{\frac{\partial Z_1(\mathbf{d})}{\partial d_i}\right\}, i \in I, \tag{2c}$$

$$\nabla_{\mathbf{d}}\mathbf{x}(\mathbf{d}) = \left\{\frac{\partial x_a(\mathbf{d})}{\partial d_i}\right\}, a \in A, i \in I, \tag{2d}$$

$$\nabla_{\mathbf{x}} Z_2(\mathbf{x}(\mathbf{d})) = \left\{\frac{\partial Z_2(\mathbf{x}(\mathbf{d}))}{\partial x_a}\right\}, a \in \widehat{A}. \tag{2e}$$

Note that the partial derivatives $\{\partial Z_1/\partial d_i\}$ and $\{\partial Z_2/\partial x_a\}$ have analytical expressions that are relatively easy to calculate. However, since the relationship between the link flow $\mathbf{x}$ and OD demand $\mathbf{d}$ is implicitly defined, it is impossible to directly represent it through mathematical

```
Step 1: Initialization: Determine the initial OD demand $\mathbf{d}^0$; set the number of iterations $l = 0$; the maximum number of iterations $l_{\max}$; and the convergence accuracy $\varepsilon_1$.
Step 2: Solve the lower-level problem: Use the improved gradient projection (iGP) algorithm [27] to solve the user equilibrium assignment to obtain the equilibrium traffic flow $\mathbf{x}^{*l}$ and calculate the upper-level objective function value $Z^l$.
Step 3: Estimate the gradient of the objective function $\nabla Z(\mathbf{d}^l)$. The descent direction is the negative gradient direction, i.e. $\mathbf{r}^l = \{r_i^l\} = -\nabla Z(\mathbf{d}^l), i \in I$. By utilizing the following equations, the nonnegative condition of the OD demand is employed to assess the necessity of adjusting the descent direction $\mathbf{r}^l$, thereby obtaining a search direction $\overline{\mathbf{r}}^l$.
$\overline{\mathbf{r}}^l = \{\overline{r}_i^l\}, i \in I$
$\overline{r}_i^l = \begin{cases} r_i^l, if\, d_i^l > 0, \text{or if } d_i^l = 0 & \text{and } r_i^l > 0, \\ 0, & \text{otherwise,} \end{cases}$
Step 4: Check the convergence condition: Determine whether the convergence condition ($\mathrm{RI} < \varepsilon_1$ or $l > l_{\max}$) is satisfied; if yes, the algorithm terminates; if no, go to Step 5.
$\mathrm{RI} = |(Z^l - Z^{l-1})/Z^{l-1}|$
Step 5: Determine the stepsize:
    (1) Determine the maximum step $\alpha_{\max}^l$, so that the adjusted OD demand all satisfy the nonnegative constraints.
    $\alpha_{\max}^l = \min\{+\infty, -d_i^l/\overline{r}_i^l: \overline{r}_i^l < 0, i \in I\}$
    (2) Search for the optimal step $\alpha^l$, so that it can minimize $Z(\mathbf{d}^l + \alpha^l \cdot \overline{\mathbf{r}}^l), \alpha^l \in [0, \alpha_{\max}^l]$.
Step 6: Update: Set $\mathbf{d}^l = \mathbf{d}^l + \alpha^l \cdot \overline{\mathbf{r}}^l$, $l = l + 1$ and go to Step 2.
```

Algorithm 1: The heuristic gradient algorithm framework for DAP.

equations. It is therefore challenging to calculate the gradient for the nonlinear implicit function $x_a(\mathbf{d})$, which is represented by the Jacobian matrix $\mathbf{J} = \{\partial x_a/\partial d_i\}$. The gradient will be calculated in this paper using both the linear and quadratic approximation approaches.

3.1. Linear Approximation. Define p_k as the proportion of traffic demand that chooses to use path k, and to facilitate the sensitivity analysis, we shall use p_k to reformulate equation (1e) as follows:

$$p_k = \frac{h_k}{d_i}, k \in K_i, i \in I, \tag{3a}$$

$$x_a = \sum_{i \in I} d_i \sum_{k \in K_i} \delta_{ak} p_k, a \in A. \tag{3b}$$

Assuming that the path flow proportion p_k of path k remains constant in the neighborhood, i.e., equation (3c) holds in the neighborhood of the current OD demand $\mathbf{d}$ and the Jacobian matrix $\mathbf{J}$ of the linear approximation can be obtained as shown in the following equation:

$$\frac{\partial x_a}{\partial d_i} = \sum_{k \in K_i} \delta_{ak} p_k, a \in A, i \in I. \tag{3c}$$

Worth noting is that the calculation of the Jacobian matrix $\mathbf{J}$ in the linear approximation depends on the path flow solution used, which means different path flow solutions from the lower user equilibrium assignment model may result in different computational results.

3.2. Quadratic Approximation. When the mapping between ODM and link flow pattern is modelled as the user equilibrium traffic assignment problem, sensitivity analysis methods are usually required to determine the rate of change in link flow as a result of OD demand fluctuation. In the literature, there are two types of sensitivity analysis methods that can be used to compute the Jacobian matrix $\mathbf{J}$. The first is Tobin and Friesz's [20] restricted network technique, which has been extensively investigated and employed since its publication. However, researchers have discovered several limitations in its application: (1) the strict complementarity condition is too strong and limits the restricted network method's application scope, (2) the method of selecting nondegenerate extreme points is not always effective, and (3) when the equilibrium solution is located at nondifferentiable points, using the restricted network method may result in incorrect results [6, 28, 29]. The second method constructs an auxiliary quadratic programming problem to compute the directional derivatives for each OD pair following the results of Patriksson [30]. When the objective function is differentiable at the current point, the exact gradient can be obtained [30]. The quadratic approximation method is more appropriate for computing large-scale traffic networks because it does not rely on direct matrix calculations and its problem structure is similar to the original equilibrium problem, allowing the equilibrium solution algorithms to be applied with minor modifications. As a result, in what follows the quadratic approximation method will be used to compute the Jacobian matrix $\mathbf{J}$, as demonstrated in Lundgren and Peterson [16]. The quadratic programming problem can be expressed in terms of current OD demand $\mathbf{d}$ under a given OD pair $\tilde{i}$ perturbed by a unit amount of change as follows:

$$\min F(\mathbf{h}') = \frac{1}{2} \cdot (\mathbf{x}')^T \cdot \nabla \mathbf{t}(\mathbf{x}^*) \cdot \mathbf{x}', \tag{4a}$$

$$\text{s.t. } \Lambda \cdot \mathbf{h}' = \boldsymbol{\rho}, \tag{4b}$$

$$\mathbf{x}' = \Delta \cdot \mathbf{h}'. \tag{4c}$$

In these equations, vector $\mathbf{h}'$ is made up of the partial derivative of the path flow h_k with respect to a unit perturbation in the OD demand $d_{\tilde{i}}$, while vector $\mathbf{x}'$ is made up of the partial derivative of the link flow x_a with respect to a unit perturbation in the OD demand $d_{\tilde{i}}$. In addition, $\Lambda = \{\Lambda_{ik}\}, k \in K_i, i \in I$ represents the OD-path incidence matrix, with a value of 1 if path k is used by OD demand $d_{\tilde{i}}$, and 0 otherwise; $\boldsymbol{\rho}$ represents the perturbation in OD demand, with all elements being zero except for element $\tilde{i}$, which is 1; $\Delta = \{\delta_{ak}\}, k \in K_i, i \in I, a \in A$ represents the link-path incidence matrix, with a value of 1 if link a is on path k, and 0 otherwise; vector $\mathbf{x}^*$ represents the equilibrium link flow under OD demand $\mathbf{d}$, and vector $\nabla\mathbf{t}(\mathbf{x}^*)$ is made up of the first derivative of the link cost $t_a(x_a^*)$ with respect to the link flow x_a^*, i.e., $\nabla t_a(x_a^*)$. This paper assumes that the link cost function is separable, then $\nabla\mathbf{t}(\mathbf{x}^*) = \{\nabla t_a(x_a^*)\}, a \in A$, the objective function (4a) can be expressed as $1/2\sum_{a\in A}\nabla t_a(x_a^*)\cdot(x_a')^2$.

The quadratic programming problem can be interpreted as a path-based user equilibrium problem in which, under the current equilibrium flow solution, an additional unit traffic demand is assigned to OD pair $\tilde{i}$, and the consequent change of cost is equal for each path in the same OD pair. The quadratic programming model differs from the original user equilibrium problem in several aspects: (1) the original link cost function is replaced by a linearized link cost function with a slope equal to the first-order derivative of the link cost function at the equilibrium solution; (2) the equilibrium path set for the quadratic programming model is fixed at the path set for the original equilibrium solution; (3) only the demand of the perturbed OD pair $\tilde{i}$ is 1, while the demand of the other OD pairs is 0; and (4) there is no nonnegative constraint since $\mathbf{h}'$ and $\mathbf{x}'$ may be positive or negative.

It should be noted that when the objective function is differentiable at the current point, i.e., the current equilibrium flow solution satisfies the strict complementary condition, the optimal solution of the quadratic programming model can be interpreted as a gradient, which is unique and independent of the choice of equilibrium path set. This differs from the linear approximation method, as different path flow solutions can yield distinct approximate gradients.

Problem (4a)–(4c) is a quadratic programming model with equation constraints that can be solved by the typical projection gradient method [31]. As demonstrated in Lundgren and Peterson [16], the projected gradient $\mathbf{w}$ of the objective function can be expressed as follows:

$$w_k = \sigma_k - \tilde{\sigma}_i, k \in K_i, i \in I, \tag{5a}$$

$$\mu_a = \nabla t_a(x_a^*)\cdot x'_a. \tag{5b}$$

Note that σ_k can be interpreted as the path cost of path k given the link cost being μ_a, i.e., $\sigma_k = \sum_{a\in A}\delta_{ak}\mu_a$, where δ_{ak} is 1 if the link a is on path k and 0 otherwise, and $\tilde{\sigma}_i$ is the average value of σ_k for all paths in OD pair i.

The steps of the algorithm for solving the quadratic program can be outlined as shown in Algorithm 2.

4. Calculation of Stepsize

In every iteration of Algorithm 1, we need a proper method to determine a stepsize, which typically plays a crucial role in ensuring the accuracy and efficacy of the whole algorithm. Next, we will discuss two distinct stepsize calculation methods that are especially tailored to solve large-scale DAP problems.

4.1. Enhanced Armijo Method. In practice, there are several methods to determine the stepsize in the gradient descent algorithms. For example, the stepsize can be determined by iteratively solving a one-dimensional optimization problem with an exact line search algorithm [32]. However, such a method may be too much expensive in our framework because it may require numerous evaluations of the objective function of the process. Therefore, the nonexact step search method named Armijo strategy [33] is employed in this study, which is based on successive reductions of the stepsize until a form of descent is achieved that guarantees convergence. The key idea of the Armijo strategy is to begin from the maximum feasible search stepsize $\alpha_{\max}^l$ that satisfies the demand's nonnegative condition, noting that l represents the iteration number of the DAP algorithm. For a given stepsize α^l, the objective function $Z(\mathbf{d}^l + \alpha^l\cdot\bar{\mathbf{r}}^l)$ is then evaluated. If the objective function value is substantially improved, i.e., if $Z(\mathbf{d}^l) - Z(\mathbf{d}^l + \alpha^l\cdot\bar{\mathbf{r}}^l) > \varepsilon_2$, where ε_2 is a predefined threshold, the search algorithm terminates; otherwise, the stepsize is reduced to $\alpha^l = \alpha^l/\theta$ based on a predefined parameter $\theta > 1$, and the objective function value is evaluated again. The procedures for applying the Armijo rule to calculate the stepsize used in Algorithm 1 is summarized as follows:

Two aspects of the algorithm described above could have a significant impact on its computational efficiency. (1) Each search iteration (c.f. Step 2.2 of Algorithm 3) requires solving a traffic assignment problem to obtain the updated link flow solution in accordance with the adjusted stepsize, which entails a significant computational burden and has a significant impact on the whole algorithm's performance in large networks. (2) A uniform stepsize (c.f. Step 2.4 of Algorithm 3) for all OD pairs does not reflect the varied potential in the descent direction of each OD pair and may severely limit the objective function's descent in the current direction due to the feasible constraints imposed by a particular OD.

In light of the above observations, we propose using the following first-order Taylor expansion method to approximately estimate the equilibrium link flow solution $\mathbf{x}^j$ at each iteration of Algorithm 3 (cf. Step 2.2 of Algorithm 3):

$$\mathbf{x}^j\left(\mathbf{d}^l + \alpha^j\cdot\bar{\mathbf{r}}^l\right) \approx \mathbf{x}\left(\mathbf{d}^l\right) + \nabla_{\mathbf{d}}\mathbf{x}\left(\mathbf{d}^l\right)\cdot\left(\alpha^j\cdot\bar{\mathbf{r}}^l\right), \tag{6}$$

where $\mathbf{x}(\mathbf{d}^l)$ is the equilibrium link flow corresponding to OD demand $\mathbf{d}^l$, and $\nabla_{\mathbf{d}}\mathbf{x}(\mathbf{d}^l)$ represents the gradient of link flow solution with respect to the OD demand at $\mathbf{d}^l$, which can be estimated using the approximate gradient method described in Section 3. Note that although the precision of the estimated $\mathbf{x}^j$ and $Z^{(l,j)}$ in each iteration of the line search

```
Step 1: Set the initial solution:
$h_k' = p_k, k \in K_{\tilde{i}}$
$h_k' = 0, k \in K_i, i \in I, i \neq \tilde{i}$
$x_a' = \sum_{k \in K_{\tilde{i}}} \delta_{ak} h_k'$
Step 2: Calculate the descent direction:
    For link $a$, the descent direction $\mu_a$ can be obtained using equation (5b), and for path $k$, the descent direction $w_k$ can be obtained using equation (5a). As a result, the descent direction of the path-based objective function is $\mathbf{y} = -\mathbf{w}$, and the descent direction of the link-based objective function is $\mathbf{z} = -\Delta \cdot \mathbf{w}$.
Step 3: Calculate the optimal stepsize:
    The optimal stepsize can be determined as follows: $\alpha = -(\mathbf{z}^T \cdot \nabla \mathbf{t}(\mathbf{x}^*) \cdot \mathbf{x}')/(\mathbf{z}^T \cdot \nabla \mathbf{t}(\mathbf{x}^*) \cdot \mathbf{z})$. Then, we update $\mathbf{h}'$ and $\mathbf{x}'$ to $\mathbf{h}' + \alpha \mathbf{y}$ and $\mathbf{x}' + \alpha \mathbf{z}$.
Step 4: Check the convergence criterion:
    Examine whether the projection gradient $\mathbf{w}$ is less than $\varepsilon$ (e.g., $\varepsilon = 10^{-4}$), terminate the algorithm if it is satisfied and go to Step 2 otherwise.
Note the optimal stepsize $\alpha$ along the descent direction for the quadratic programming model in Step 3 can be obtained by solving: $\min_\alpha 1/2(\mathbf{x}' + \alpha \mathbf{z})^T \cdot \nabla \mathbf{t}(\mathbf{x}^*) \cdot (\mathbf{x}' + \alpha \mathbf{z})$.
```

Algorithm 2: Algorithm for solving the quadratic programming model.

algorithm may be decreased to some extent by employing this approximation method, the convergence efficiency of the overall algorithm could be significantly enhanced in solving large-scale problems, as demonstrated by the numerical experiments in Section 5.

The second aspect of the concerns may be addressed by computing a distinct stepsize α_i^l for each OD pair $i \in I$, i.e., a stepsize vector $\boldsymbol{\alpha}^l = \{\alpha_i^l\}, i \in I$ for Algorithm 1. This can be realized by simply defining a separate maximum feasible stepsize $\alpha_{\max}^l i$ for each OD pair and then applying the Armijo strategy to ensure a significant decrease in the objective. To this end, we define a diagonal matrix $\boldsymbol{\alpha}_{\max}^l$ with its dimensions equal to the number of OD pairs, and the maximum feasible stepsize $\alpha_{\max}^l i$ for OD pair i is calculated by

$$\alpha_{\max}^l i = \begin{cases} \min\{\gamma, -d_i^l/\overline{r}_i^l\}, & \overline{r}_i^l < 0, \\ \gamma, & \overline{r}_i^l > 0. \end{cases}, i \in I, \tag{7a}$$

where γ is the preset maximum step parameter, e.g., $\gamma = 100$. Accordingly, the objective function, the trail stepsize in Algorithm 3, and formulation (6) can be updated accordingly to

$$\boldsymbol{\alpha}^0 = \boldsymbol{\alpha}_{\max}^l, \tag{7b}$$

$$Z^{(l,j)} = Z\left(\mathbf{d}^l + \boldsymbol{\alpha}^j \cdot \overline{\mathbf{r}}^l\right), \tag{7c}$$

$$\boldsymbol{\alpha}^j = \boldsymbol{\alpha}^j \cdot \boldsymbol{\theta}, \tag{7d}$$

$$\mathbf{x}^j\left(\mathbf{d}^l + \boldsymbol{\alpha}^j \cdot \overline{\mathbf{r}}^l\right) \approx \mathbf{x}\left(\mathbf{d}^l\right) + \nabla_{\mathbf{d}} \mathbf{x}\left(\mathbf{d}^l\right) \cdot \left(\boldsymbol{\alpha}^j \cdot \overline{\mathbf{r}}^l\right), \tag{7e}$$

where $\boldsymbol{\alpha}^0$, $\boldsymbol{\alpha}^j$, and $\boldsymbol{\theta}$ are diagonal matrices with dimensions corresponding to the number of OD pairs. All diagonal elements of $\boldsymbol{\theta}$ are equal to $1/\theta$. Furthermore, $\overline{\mathbf{r}}^l$ denotes the search direction defined in Algorithm 1.

As a result, the Armijo strategy presented in Algorithm 3 can be enhanced by incorporating the above two improvement methods, leading to an enhanced Armijo method described in Algorithm 4.

The update formula in Step 6 of the corresponding Algorithm 1 is replaced with

$$\mathbf{d}^l = \mathbf{d}^l + \boldsymbol{\alpha}^l \cdot \overline{\mathbf{r}}^l. \tag{8}$$

It is anticipated that the enhanced Armijo method will outperform the Armijo method in terms of both computational efficiency and solution precision. First, it is evident that using the first-order Taylor approximation rather than the user equilibrium assignment to update flow solutions could save a substantial amount of computational time at the expense of a certain degree of accuracy loss. Second, the use of the OD-specific stepsize strategy can partially mitigate for the loss of precision caused by the first-order Taylor approximation, resulting in a generally improved convergence result.

4.2. Enhanced Analytical Calculation Method.

Due to the presence of a nonlinear implicit function $\mathbf{x}(\mathbf{d})$ in the objective function, an analytical formula for calculating the optimal stepsize cannot be established. Nevertheless, it is straightforward to derive an analytical expression for calculating the optimal stepsize if we approximate the implicit function $\mathbf{x}(\mathbf{d})$ using the following formulation:

$$\mathbf{x}\left(\mathbf{d}^l + \lambda \cdot \overline{\mathbf{r}}^l\right) \approx \mathbf{x}\left(\mathbf{d}^l\right) + \nabla_{\mathbf{d}} \mathbf{x}\left(\mathbf{d}^l\right) \cdot \left(\lambda \cdot \overline{\mathbf{r}}^l\right). \tag{9}$$

Accordingly, the optimal stepsize λ, given the demand $\mathbf{d}^l$ and search direction $\overline{\mathbf{r}}^l$, can be obtained by solving the following one-dimensional subproblem:

$$\begin{aligned}\min_\lambda Z\left(\mathbf{d}^l + \lambda \cdot \overline{\mathbf{r}}^l\right) = {} & \theta_1 \cdot \left(\mathbf{d}^l + \lambda \cdot \overline{\mathbf{r}}^l - \widehat{\mathbf{d}}\right)^T \cdot \left(\mathbf{d}^l + \lambda \cdot \overline{\mathbf{r}}^l - \widehat{\mathbf{d}}\right) \\ & + \theta_2 \cdot \left(\mathbf{x}\left(\mathbf{d}^l + \lambda \cdot \overline{\mathbf{r}}^l\right) - \widehat{\mathbf{x}}\right)^T \\ & \cdot \left(\mathbf{x}\left(\mathbf{d}^l + \lambda \cdot \overline{\mathbf{r}}^l\right) - \widehat{\mathbf{x}}\right).\end{aligned} \tag{10a}$$

Note that the subproblem (10a) can be reformulated as follows by incorporating formulation (9) into it.

```
Step 1: Determine the maximum feasible stepsize $\alpha^l_{\max}$ according to Algorithm 1 such that the adjusted OD demand satisfies the nonnegative constraints.
Step 2: Search for an appropriate stepsize $\alpha^l$ according to the Armijo rule:
    2.1 Set $\alpha^0 = \alpha^l_{\max}$, the search iteration number $j = 0$, the maximum search iteration number $j_{\max}$, and $Z^{(l,0)} = Z(\mathbf{d}^l)$.
    2.2 Obtain the updated equilibrium flow solution $\mathbf{x}^j$ pertaining to the adjusted OD demand by employing user equilibrium assignment, and calculate the corresponding value of the upper-level objective function:
$Z^{(l,j)} = Z(\mathbf{d}^l + \alpha^j \cdot \overline{\mathbf{r}}^l)$
Note that $\overline{\mathbf{r}}^l$ denotes the search direction defined in Algorithm 1.
    2.3 If the convergence condition $Z^{(l,0)} - Z^{(l,j)} > \varepsilon_2$ is satisfied, set $\alpha^l = \alpha^j$ and stop the algorithm; otherwise, proceed to 2.4.
    2.4 If $j$ is equal to $j_{\max}$, set $\alpha^l = \operatorname{argmin}_{\alpha^j}(Z^{(l,j)})$; otherwise, set $\alpha^j = \alpha^j/\theta$, $j = j + 1$, and go to Step 2.
```

Algorithm 3: Armijo strategy used in DAP algorithm.

```
Step 1: Determine the maximum feasible stepsize $\boldsymbol{\alpha}^l_{\max}$ according to equation (7a) such that the adjusted OD demand satisfies the nonnegative constraints.
Step 2: Search for an appropriate stepsize $\boldsymbol{\alpha}^l$, according to the Enhanced Armijo method:
    2.1 Set $\boldsymbol{\alpha}^0 = \boldsymbol{\alpha}^l_{\max}$, the search iteration number $j = 0$, the maximum search iteration number $j_{\max}$, and $Z^{(l,0)} = Z(\mathbf{d}^l)$.
    2.2 Estimate the updated flow solution $\mathbf{x}^j$ pertaining to the adjusted OD demand by employing equation (7e), and calculate the corresponding value of the upper-level objective function using equation (7c).
    2.3 If the convergence condition $Z^{(l,0)} - Z^{(l,j)} > \varepsilon_2$ is satisfied, set $\boldsymbol{\alpha}^l = \boldsymbol{\alpha}^j$ and stop the algorithm; otherwise, proceed to 2.4.
    2.4 If $j$ is equal to $j_{\max}$, set $\boldsymbol{\alpha}^l = \operatorname{argmin}_{\boldsymbol{\alpha}^j}(Z^{(l,j)})$; otherwise, set $\boldsymbol{\alpha}^j = \boldsymbol{\alpha}^j \cdot \boldsymbol{\theta}$, $j = j + 1$, and go to Step 2.
```

Algorithm 4: Enhanced Armijo method used in DAP algorithm.

$$\min_\lambda \theta_1 \cdot \left(\mathbf{d}^l + \lambda \cdot \overline{\mathbf{r}}^l - \widehat{\mathbf{d}}\right)^T \cdot \left(\mathbf{d}^l + \lambda \cdot \overline{\mathbf{r}}^l - \widehat{\mathbf{d}}\right) + \theta_2 \cdot \left(\mathbf{x}\left(\mathbf{d}^l\right) + \nabla_{\mathbf{d}}\mathbf{x}\left(\mathbf{d}^l\right) \cdot \left(\lambda \cdot \overline{\mathbf{r}}^l\right) - \widehat{\mathbf{x}}\right)^T \cdot \left(\mathbf{x}\left(\mathbf{d}^l\right) + \nabla_{\mathbf{d}}\mathbf{x}\left(\mathbf{d}^l\right) \cdot \left(\lambda \cdot \overline{\mathbf{r}}^l\right) - \widehat{\mathbf{x}}\right). \tag{10b}$$

For the sake of simplicity, the superscript l is omitted for brevity, and the algebraic form of equation (10b) is given in the following equation:

$$\min_\lambda \ \theta_1 \cdot \sum_{i\in I} \left(d_i + \lambda \cdot \overline{r}_i - \widehat{d}_i\right)^2 + \theta_2 \cdot \sum_{a\in \widehat{A}} \left(x_a(\mathbf{d}) + \sum_{i\in I} \frac{\partial x_a}{\partial d_i} \cdot \overline{r}_i \cdot \lambda - \widehat{x}_a\right)^2, \tag{10c}$$

where $x_a(\mathbf{d})$ represents the flow on link a corresponding to the OD demand $\mathbf{d}$. Considering $(\partial x_a/\partial d_i) \cdot \overline{r}_i$ as φ_i^a, that is, $\varphi_i^a = (\partial x_a/\partial d_i)\overline{r}_i, i \in I, a \in \widehat{A}$, then the optimal stepsize λ is given by the following equation:

$$\lambda = \frac{\theta_1 \cdot \sum_{i\in I}\left(\widehat{d}_i - d_i\right) \cdot \overline{r}_i + \theta_2 \cdot \sum_{a\in\widehat{A}}\left[\left(\widehat{x}_a - x_a\right) \cdot \sum_{i\in I}\varphi_i^a\right]}{\theta_1 \cdot \sum_{i\in I}\overline{r}_i^{\,2} + \theta_2 \cdot \sum_{a\in\widehat{A}}\left(\sum_{i\in I}\varphi_i^a\right)^2}. \tag{11}$$

Importantly, a feasibility test must be performed on the above stepsize to ascertain whether or not the nonnegative OD demand constraint is satisfied. If the constraint is met, the stepsize computed by equation (11) is considered feasible. If the feasible constraint is not met, however, the enhanced Armijo method will be used to determine an alternative stepsize for OD pairs that do not satisfy the demand's nonnegative requirement. Although the analytical calculation method may require modification with the enhanced Armijo algorithm in some cases, it still offers significant advantages in stepsize searching due to its simple mathematical formula that eliminates the need for repeated evaluations of the objective function, thereby reducing the computational burden of stepsize searching for large-scale networks.

Note that Spiess [13] also developed an analytical formula for calculating the optimal stepsize, which we refer to as the Spiess' method hereafter. Following are the differences between the two approaches: (1) Spiess' method is predominantly based on the linear approximation method described in Section 3.1 to estimate $\nabla_{\mathbf{d}}\mathbf{x}(\mathbf{d})$ and assumes a constant path flow proportion p_k within the neighborhood of $\mathbf{d}$. In contrast, our method makes no assumptions regarding its dependence on the linear method and can also be used to solve the case where the descent direction is estimated by the quadratic approximation method presented in

Section 3.2, which tends to produce a more accurate descent direction than the linear approximation method. (2) Spiess' method does not specify how to choose a feasible stepsize when the calculated one does not satisfy the nonnegative OD demand constraints. Our method proposes using the enhanced Armijo method as a workaround when the nonnegative constraint is not satisfied and assures that the obtained descent direction is certainly feasible.

The implementation of the proposed analytical calculation method is described in Algorithm 5, and several points are worth noting: first, the maximum feasible stepsize matrix $\boldsymbol{\alpha}^l_{\max}$ is initially specified as follows: $\alpha^l_{\max}i = \lambda$ if it satisfies the nonnegative constraint for OD pair i; otherwise, $\alpha^l_{\max}i$ is set according to the formula (7a). Second, the tentative stepsize matrix $\boldsymbol{\alpha}^j$ is updated by $\boldsymbol{\alpha}^j = \boldsymbol{\alpha}^j \cdot \boldsymbol{\vartheta}$, where $\boldsymbol{\vartheta}$ is a diagonal matrix with dimensions equal to the number of OD pairs, and the diagonal elements of $\boldsymbol{\vartheta}$ are set to 1 for the OD pairs that meet the nonnegative constraints, and $1/\theta$ for those not satisfying the constraint.

5. Numerical Results

This section is organized into three parts. The first describes the algorithm computing environment, test networks, and algorithm implementation details. The performance of the test algorithms in solving medium networks is compared in the second part. The last part further examines the computation time and the convergence magnitude with each iteration of the algorithms for large networks.

5.1. Computing Environment and Algorithm Implementation Details. All algorithms are coded using the Toolkit of Network Modeling, a C++ class library specialized in modelling transportation networks [34]. All numerical results reported in this section were produced on a Windows 10 64-bit PC with Intel® Core™ i5-7300HQ CPU 2.50 GHz and 16G RAM. The convergence performance of the test algorithms is evaluated by applying them to solving the Sioux-Falls and Chicago-Sketch networks, available at http://www.bgu.ac.il/?bargera/tntp/, with the topology of each network described in Table 1, using the Bureau of Public Roads (BPR) function with a coefficient of 0.15 and a degree of 4 to model the link travel time. The parameter settings used in numerical experiments are presented in Table 2.

This study will implement and compare a total of six algorithms for DAP that differ substantially in their selection of methods for gradient approximation and stepsize calculation. Two of these algorithms have been documented in the literature and were developed by Spiess [13] and Lundgren and Peterson [16], respectively. In particular, Spiess' algorithm employs the linear approximation method for determining the descent direction and the standard analytical-calculation (sAC) method for calculating the stepsize. Using the quadratic approximation method and the standard Armijo method, Lundgren's algorithm prioritizes descent direction precision and solution quality. The other four algorithms are developed based on the new stepsize computation methods proposed in this study, and they are as follows: quadratic-approximation and enhanced-Armijo (QaEA) algorithm, quadratic-approximation and enhanced-analytical-calculation (QaEAC) algorithm, linear-approximation and enhanced-Armijo (LaEA) algorithm, and linear-approximation and enhanced-analytical-calculation (LaEAC) algorithm. The algorithms of Spiess and Lundgren will serve as benchmarks for evaluating the performance of the above four algorithms in terms of solution accuracy and efficiency.

A few details of the strategies used in our implementation of the algorithms are given below. (1) It is computationally inefficient to consider the interdependence of all network paths when using Algorithm 2 to solve the quadratic programming model, especially for large-scale networks where it can lead to an excessively accurate search direction, demanding substantial CPU time and memory resources. So, we only consider the paths that belong to the same OD pair. Equation (5a) therefore only considers the paths related to the perturbed OD pair $\tilde{i}$, i.e., $k \in K_{\tilde{i}}$. Although this method may not make a very accurate calculation, it permits more efficient updates and evaluations of new search directions, which is especially advantageous for Algorithm 1. A detailed discussion can be found in Lundgren and Peterson [16]. (2) The improved gradient projection (iGP) algorithm [27] is used to solve the standard user equilibrium traffic assignment subproblem at each main iteration. Note that the computation of the search direction depends on the accuracy of the solution to the assignment problem. Furthermore, Boyce et al. [35] argued that a relative gap of 0.0001 is sufficient for solving assignment problems. Therefore, the convergence criteria of the traffic assignment subproblem is used as a relative gap of 10^{-7} to ensure the precise solution of the search direction. (3) The historical OD demand matrix is taken as the initial OD demand matrix, and approximately thirty percent of network links are chosen at random to function as the observed links, designated as $\hat{A}$. The observed link flows are generated by perturbing the historical OD demand matrix $\hat{\mathbf{d}}$ at random and then executing the user equilibrium traffic assignment, as shown in the following equation:

$$\hat{x}_a = \hat{x}_a(\hat{\mathbf{d}} \cdot (1 + \epsilon)), \forall a \in \hat{A}, \tag{12}$$

where ϵ is the random number, $\epsilon \in [-0.3,0.3]$.

5.2. Sioux-Falls Network Analysis. In the first set of experiments, we compared the effectiveness of EA-class (i.e., LaEA and QaEA) and EAC-class (i.e., LaEAC and QaEAC) algorithms with two benchmark methods for the Sioux Falls network (Figure 1). Note that it is challenging to estimate the burden associated with a single iteration of each algorithm due to the constant range of 1 second for the termination time of the tested algorithms. As a consequence, it becomes difficult to identify a distinct computational advantage among them. Therefore, we have decided to refrain from discussing algorithmic computational efficiency in this context and instead focus on algorithmic solution accuracy disparities.

```
Step 1: Determine the approximated calculate optimal stepsize $\lambda$, according to equation (11).
Step 2: If the nonnegative condition of the calculate stepsize for OD demand is satisfied, set diagonal elements of matrices $\boldsymbol{\alpha}^l$ are $\lambda$;
otherwise, proceed to Step 3.
Step 3: Search for an appropriate stepsize $\boldsymbol{\alpha}^l$ according to the Enhanced Armijo method.
```

Algorithm 5: Enhanced analytical calculation method used in DAP algorithm.

Table 1: Profile information of test networks used in numerical experiments.

Networks	Size	Nodes	Links	OD pairs number
Sioux-Falls	Medium size	24	76	528
Chicago-Sketch	Large	933	2950	93513

Table 2: Parameter settings used in numerical experiments.

θ_1	θ_2	θ	$j_{\max}$	ε_1	$l_{\max}$
1	1	10	5	10^{-5}	40

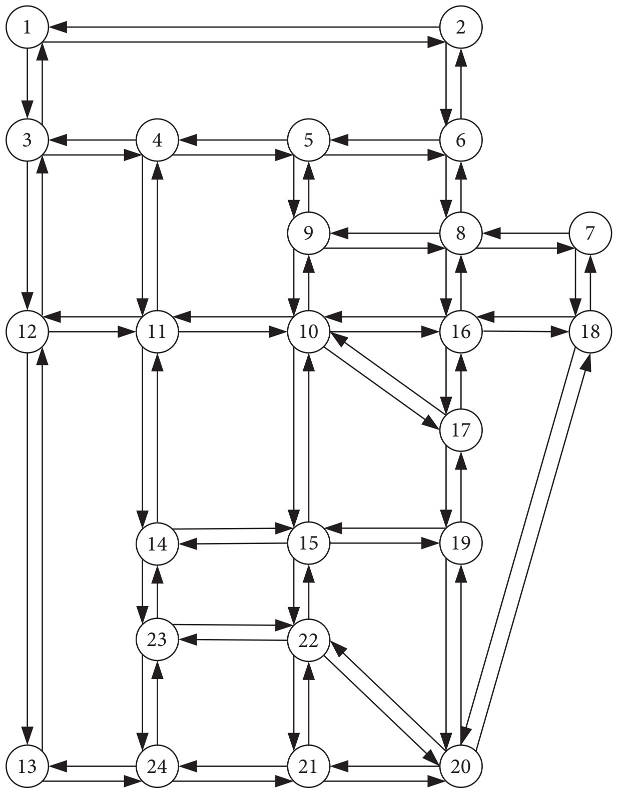


Figure 1: Topology of the Sioux-Falls network.

The computational results of the test algorithms on the Sioux-Falls network are shown in Figure 2. In these plots, the objective value and $\log_{10}$ (RI) are represented on the vertical axis, while the horizontal axis denotes the number of iterations. Overall, the test network shows that the proposed QaEAC and QaEA algorithms demonstrate a certain advantage over Lundgren-08 and Spiess-90. When achieving the desired accuracy, the QaEAC algorithm performs 5.3% lower than Lundgren-08 and 9.8% lower than Spiess-90 in terms of objective values. When the QaEA algorithm reaches the maximum number of iterations, it achieves an objective value that is 4.1% lower than Lundgren-08 and 8.6% lower than Spiess-90. Despite the seeming smallness of these changes, they demonstrate that the QaEAC and QaEA algorithms are capable of searching for better solutions than the benchmark algorithms when the stepsize calculation approach is properly taken into account. It is worth noting that while LaEAC and LaEA obtain inferior solutions compared to Lundgren-08 and Spiess-90, both exhibit significant advantages in terms of computational efficiency, which will be further validated in large-scale networks.

The test algorithms are carefully compared in the sections that follow, and we examine how alternative approaches to computing stepsize affect how well the algorithms solve problems. We evaluate the algorithms under identical iteration counts and the gradient approximation method to provide a fair comparison.

We further introduced the quadratic-approximation and standard-analytical-calculation (QasAC) methods to assess the differences in solution performance between EA-class algorithms and standard AC-class algorithms. The computational results of QaEA, Lundgren-08, and QasAC on the Sioux-Falls network are illustrated in Figure 3. Despite the fact that QaEA estimates equilibrium flow solutions using a first-order Taylor approximation rather than the user equilibrium assignment, it routinely surpasses Lundgren-08 and QasAC in terms of solution quality. In particular, QaEA reduces the objective value by 4.1% when compared to Lundgren-08 and 0.4% when compared to QasAC. This is due to the fact that EA-class algorithms use an OD-specific stepsize strategy to determine a distinct stepsize for each OD pair, which helps to offset the accuracy loss associated with approximation equilibrium solutions. The EA-class algorithms routinely beat the standard Armijo-class and standard AC-class algorithms in terms of solution accuracy by properly predicting equilibrium flow solutions using quadratic programming to generate the Jacobian matrix $\mathbf{J} = \{\partial x_a / \partial d_i\}, a \in A, i \in I$. To corroborate this argument, we conducted tests in which we removed the OD-specific stepsize strategy inside the framework of the EA-class algorithms, resulting in the EAs-class algorithm, a simplified EA-class algorithm. Figure 4 depicts the computational results of QaEA and QaEAs on the Sioux-Falls network. It is observed that the solution performance of the EAs-class algorithm is inferior to that of the EA-class algorithm, with a 6.0% increase in the objective function value. Therefore, it can be confidently stated that the proposed OD-specific stepsize strategy contributes positively to enhancing the solution accuracy of the EA-class algorithms.

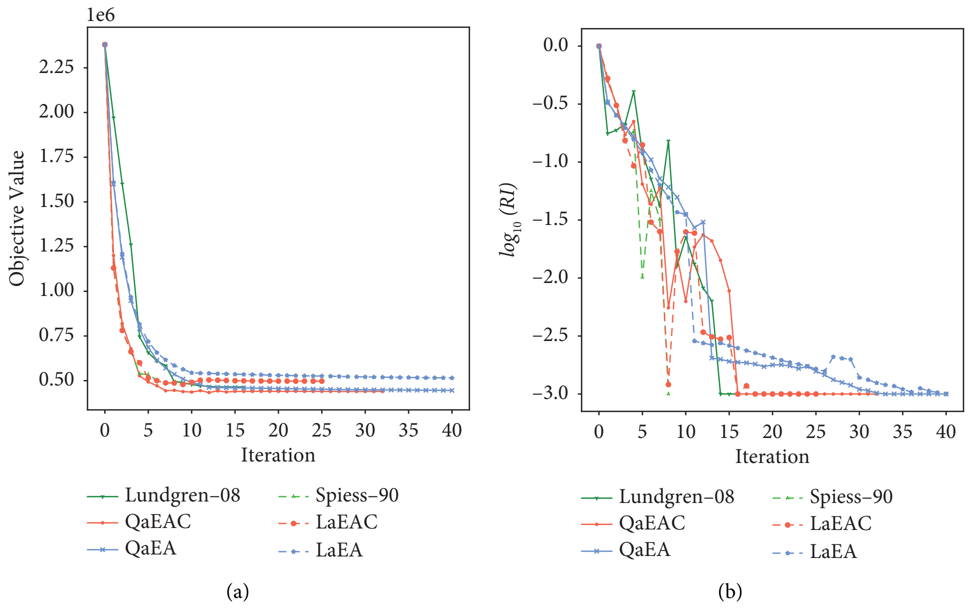


Figure 2: Convergence performance comparison on the Sioux-Falls network.

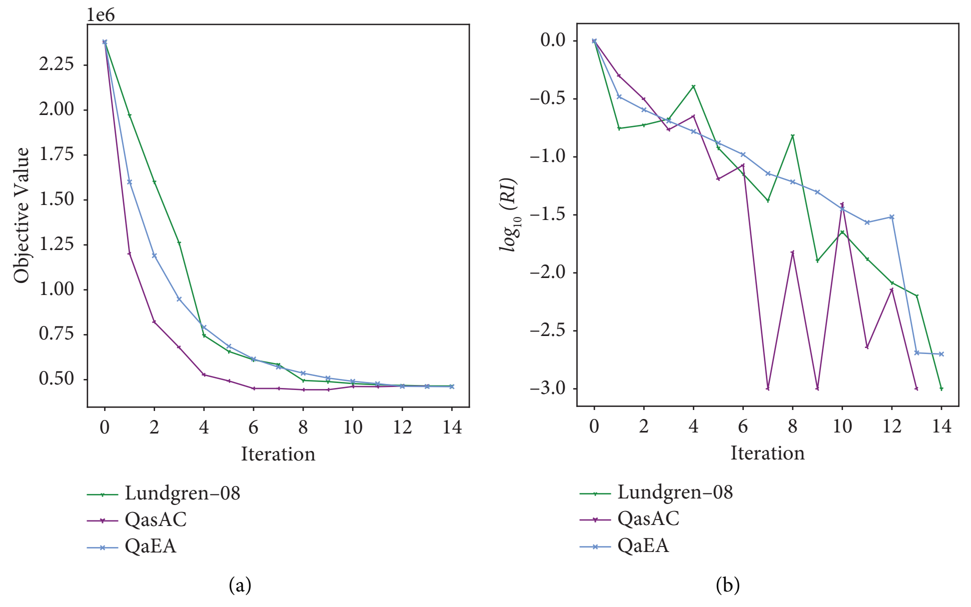


Figure 3: Comparison on the convergence performance of Lundgren-08, QasAC and QaEA.

Figures 5 and 6 compare the computational performance of the EAC-class and the EA-class algorithms for the Sioux Falls network. LaEAC obtains a 5.3% reduction in objective value compared to LaEA, while QaEAC achieves a 1.9% reduction compared to QaEA. This is because, unlike the EA-class algorithms, which use a trial-and-error method to determine a feasible stepsize, the EAC-class algorithms generate an approximately optimal formulation for the stepsize, which tends to offer greater computational accuracy if the demand feasible constraint is fulfilled.

In conclusion, the EA-class algorithms beat the benchmark algorithms in terms of computation accuracy, and the EAC-class algorithms improve the performance of the EA-class algorithms even further. These findings will be validated further in large-scale networks.

5.3. Chicago-Sketch Network Analysis. In the second set of tests, we employed a network model named Chicago-Sketch (Figure 7). Since the Chicago-Sketch network is a sizable metropolitan network, we were able to easily estimate the computational effort associated with each iteration using CPU time, enabling us to assess the computational effectiveness of various methods. As a result, we evaluated the performance of proposed algorithms in this set of tests from the perspectives of computational efficiency and solution accuracy.

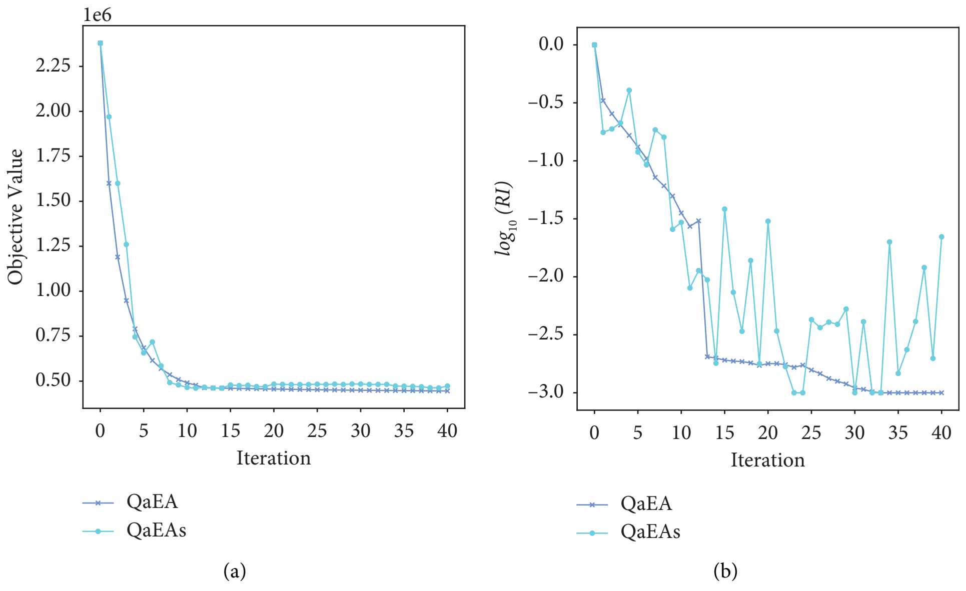


Figure 4: Comparison on the convergence performance of QaEA and QaEAs.

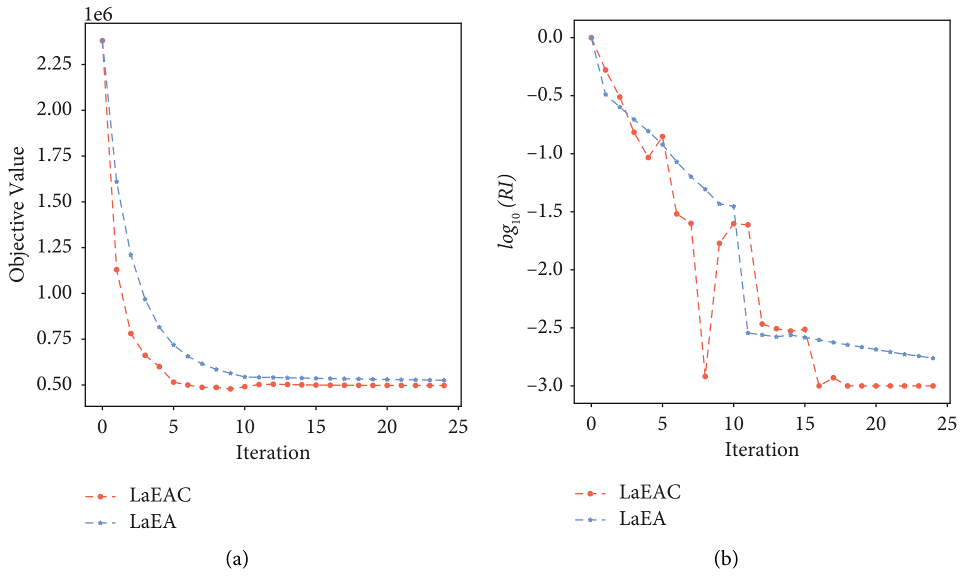
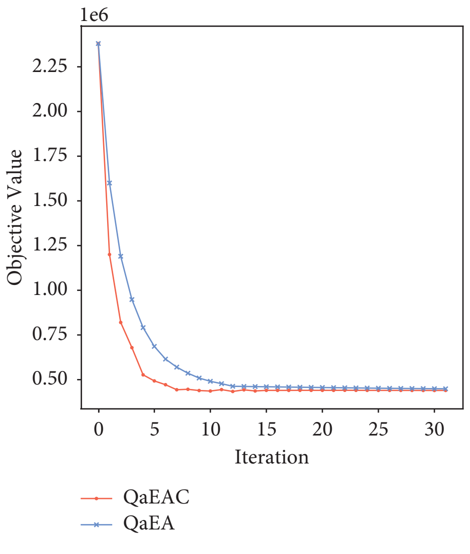


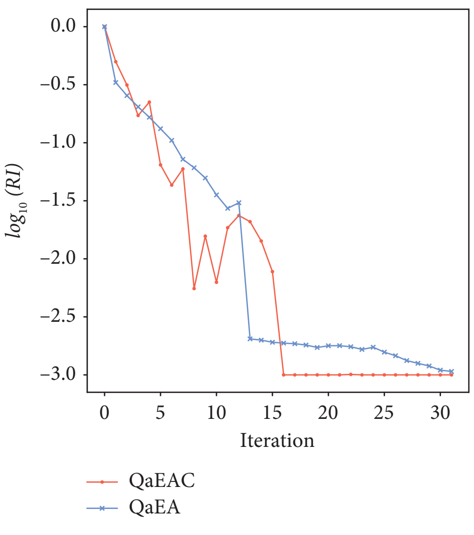
Figure 5: Comparison on the convergence performance of LaEAC and LaEA.

Figure 8 plots the computational results of the test algorithms on the Chicago-Sketch network. We concentrate on two metrics: the algorithm's minimal objective function value at termination and the total CPU time required for achieving that value. Overall, EAC-class algorithms can achieve better results at lower computing costs than benchmark algorithms for large-scale networks. QaEAC, in particular, decreases overall CPU time by 20.6% when compared to Lundgren-08 while obtaining a 49.3% lower objective value. When compared to Spiess-90, LaEAC reduces overall CPU time by 18.8% while achieving a 50.9% lower objective value. EA-class algorithms, on the other hand, offer greater advantages in computational efficiency as compared to benchmark algorithms, but their solution accuracy is dependent on the computational correctness of the search direction. For example, as compared to Lundgren-08, QaEA reduces overall CPU time by 25.1% while obtaining a 13.1% lower objective value. Meanwhile, LaEA decreases overall CPU time by 27.8% when compared to Spiess-90, but the objective value increases by 50.7%, owing to a less precise estimate of the search direction, resulting in an inferior solution. These findings are consistent with previously reported results.

Figure 9 shows the computation time per iteration of the test algorithms. Both EA-class and EAC-class algorithms exhibit significantly shorter computational time per iteration compared to the benchmark algorithms, while the EAC-class algorithm shows slightly higher computational time



(a)



(b)

Figure 6: Comparison on the convergence performance of QaEAC and QaEA.

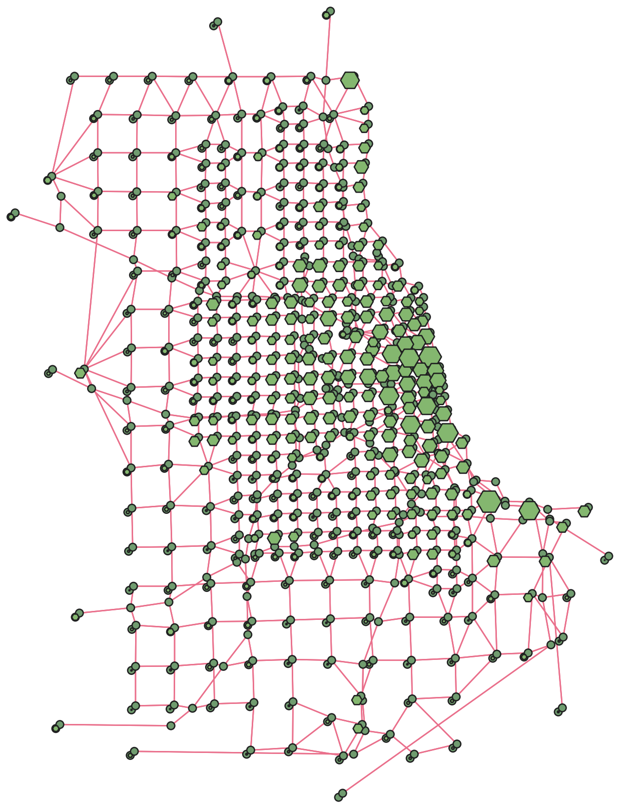


Figure 7: Topology of the Chicago-Sketch network.

compared to the EA-class algorithm, which is consistent with the complexity analysis results.

It is worth noting that the computational efficiency of EAC-class algorithms is not necessarily always lower than that of EA-class algorithms. This is dependent on whether the nonnegative constraint is met by the approximately optimal stepsize that the EAC-class attained in each iteration. If so, then EAC-class algorithms take less time to compute each iteration than EA-class algorithms. To find stepsize that satisfy the nonnegative criterion of OD demand, EAC-class must continue performing EA-class if the condition is not satisfied, adding acceptable time to the calculation process.

The computational results of LaEAC compared to QasAC and Spiess-90 on the Chicago-Sketch network are shown in Figure 10. Among them, LaEAC and Spiess-90 employ the Linear approximation approach, while QasAC and Spiess-90 utilize the sAC computing stepsize method. Overall, LaEAC outperforms Spiess-90 and QasAC by obtaining superior solutions in a lower calculation time. In particular, LaEAC reduces objective value by 51.0% when compared to Spiess-90, and the overall CPU time is reduced by 18.8% as well. In comparison, QasAC reduces objective value by 2.3% when compared to Spiess-90, but overall CPU time increases noticeably by 68.6% due to the more accurate search direction. This gives rise to the idea that, in addition to improving the search direction method, enhancing the computing stepsize approach is also crucial for boosting algorithm performance. This makes sense because the performance of the algorithm is greatly impacted by the choice of the stepsize methods in addition to the algorithm's accuracy of the search direction.

In conclusion, EA-class algorithms have shown significant improvements in both solution effectiveness and computational efficiency compared to benchmark algorithms. On the other hand, EAC-class algorithms offer a significant advancement over EA-class algorithms since they can produce better results at comparable computing costs, which makes them especially well-suited for large-scale networks.

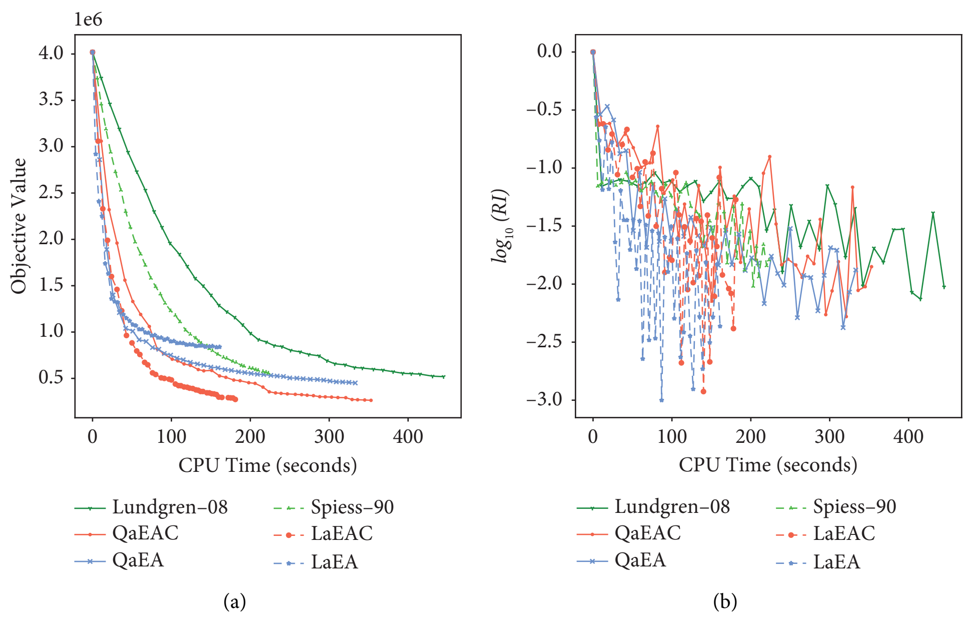


Figure 8: Convergence performance comparison on the Chicago-Sketch network.

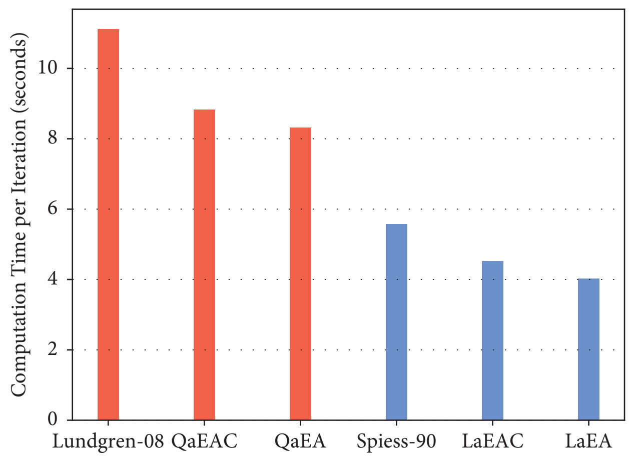


Figure 9: Average computation time per iteration for test algorithms.

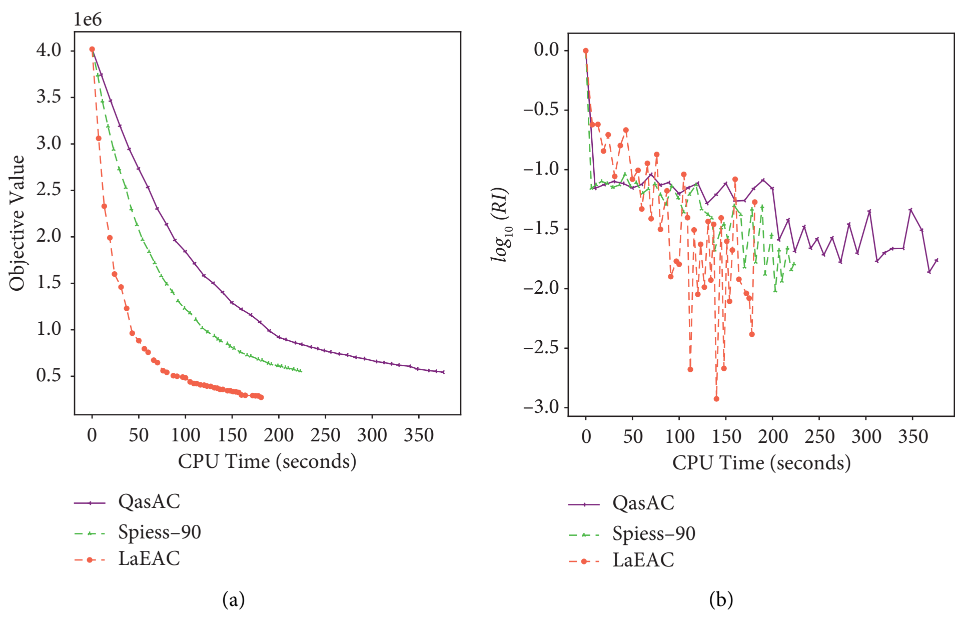


FIGURE 10: Comparison on the convergence performance of QasAC, Spiess-90, and LaEAC.

6. Conclusion

We present a framework for heuristic gradient algorithms that identifies commonalities between the majority of approximate gradient algorithms. This algorithm framework has been investigated in the past, but its efficient implementation and numerical evidence from real-world applications remain uncommon. We investigate and contrast heuristic gradient methods from both analytic and numerical perspectives within this framework.

The framework also permits a precise comparison of the effects of various gradient approximation and stepsize calculation methods on the overall performance of the algorithm. By combining two newly developed stepsize calculation methods with two categories of approximate gradient methods, we contribute several new algorithms to the DAP algorithm family. The similarities and differences between these algorithms were also covered in great detail. Numerical experiments demonstrate that algorithms using the new stepsize calculation strategies consistently beat existing algorithms in terms of computational accuracy and efficiency. This finding highlights the significant impact of the stepsize calculation method on algorithm performance.

As the potential of the proposed algorithms in confronting the DAP problem has been demonstrated in this paper, future research could focus on solving DAP models with more complex equilibrium models, such as the stochastic, multiclass, or asymmetric user equilibrium assignment models.

Data Availability

All data included in this study are available upon request by contact with the corresponding author.

Disclosure

However, the funding agency had no involvement in the study design, data collection, analysis, interpretation of results, or writing of the manuscript.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (71971178, 72371205, and 52232011); Chengdu Science and Technology Program (Grant 2021-YF05-00177-SN, 2021-RK00-00067-ZF); and Sichuan Science and Technology Program (Grant no. 2021YFH0041).

References

[1] Y. Sheffi, *Urban Transportation Networks*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1985.

[2] H. D. Sherali, R. Sivanandan, and A. G. Hobeika, “A linear programming approach for synthesizing origin-destination trip tables from link traffic volumes,” *Transportation Research Part B: Methodological*, vol. 28, no. 3, pp. 213–233, 1994.

[3] D. E. Low, “New approach to transportation systems modeling,” *Traffic quarterly*, vol. 26, no. 3, 1972.

[4] T. Abrahamsson, *Estimation of Origin-Destination Matrices Using Traffic Counts-A Literature Survey*, IIASA, Laxenburg, Austria, 1998.

[5] S. Bera and K. V. Rao, “Estimation of origin-destination matrix from traffic counts: the state of the art,” *European Transport/Trasporti Europei*, vol. 49, 2011.

[6] M. G. H. Bell and Y. Iida, *Transportation Network Analysis*, Wiley, Hoboken, NJ, USA, 1997.

[7] E. Codina and L. Montero, "Approximation of the steepest descent direction for the OD matrix adjustment problem," *Annals of Operations Research*, vol. 144, no. 1, pp. 329–362, 2006.

[8] H. J. Van Zuylen and L. G. Willumsen, "The most likely trip matrix estimated from traffic counts," *Transportation Research Part B: Methodological*, vol. 14, no. 3, pp. 281–293, 1980.

[9] M. Carey, C. Hendrickson, and K. Siddharthan, "A method for direct estimation of origin/destination trip matrices," *Transportation Science*, vol. 15, no. 1, pp. 32–49, 1981.

[10] H. Spiess, "A maximum likelihood model for estimating origin-destination matrices," *Transportation Research Part B: Methodological*, vol. 21, no. 5, pp. 395–412, 1987.

[11] M. Brenninger-Göthe, K. O. Jörnsten, and J. T. Lundgren, "Estimation of origin-destination matrices from traffic counts using multiobjective programming formulations," *Transportation Research Part B: Methodological*, vol. 23, no. 4, pp. 257–269, 1989.

[12] M. Bierlaire and P. L. Toint, "Meuse: an origin-destination matrix estimator that exploits structure," *Transportation Research Part B: Methodological*, vol. 29, no. 1, pp. 47–60, 1995.

[13] H. Spiess, *A Gradient Approach for the OD Matrix Adjustment Problem*, Centre de Recherche sur les Transports, Universite´ de Montreál, Montreal, Canada, 1990.

[14] H. Yang, T. Sasaki, Y. Iida, and Y. Asakura, "Estimation of origin-destination matrices from link traffic counts on congested networks," *Transportation Research Part B: Methodological*, vol. 26, no. 6, pp. 417–434, 1992.

[15] E. Codina, R. García, and A. Marín, "New algorithmic alternatives for the O–D matrix adjustment problem on traffic networks," *European Journal of Operational Research*, vol. 175, no. 3, pp. 1484–1500, 2006.

[16] J. T. Lundgren and A. Peterson, "A heuristic for the bilevel origin–destination-matrix estimation problem," *Transportation Research Part B: Methodological*, vol. 42, no. 4, pp. 339–354, 2008.

[17] L. Vicente, G. Savard, and J. Júdice, "Descent approaches for quadratic bilevel programming," *Journal of Optimization Theory and Applications*, vol. 81, no. 2, pp. 379–399, 1994.

[18] J. Walpen, E. M. Mancinelli, and P. A. Lotito, "A heuristic for the OD matrix adjustment problem in a congested transport network," *European Journal of Operational Research*, vol. 242, no. 3, pp. 807–819, 2015.

[19] H. Yang, "Heuristic algorithms for the bilevel origin-destination matrix estimation problem," *Transportation Research Part B: Methodological*, vol. 29, no. 4, pp. 231–242, 1995.

[20] R. L. Tobin and T. L. Friesz, "Sensitivity analysis for equilibrium network flow," *Transportation Science*, vol. 22, no. 4, pp. 242–250, 1988.

[21] Y. Chen and M. Florian, "OD demand adjustment problem with congestion: Part I. model analysis and optimality conditions," *Advanced Methods in Transportation Analysis*, Springer Berlin Heidelberg, Berlin, Germanypp. 1–27, 1996.

[22] A. Chen, X. Xu, S. Ryu, and Z. Zhou, "A self-adaptive Armijo stepsize strategy with application to traffic assignment models and algorithms," *Transportmetrica: Transportation Science*, vol. 9, no. 8, pp. 695–712, 2013.

[23] M. G. H. Bell, "The estimation of origin-destination matrices by constrained generalised least squares," *Transportation Research Part B: Methodological*, vol. 25, no. 1, pp. 13–22, 1991.

[24] M. Carey and R. Revelli, "Constrained estimation of direct demand functions and trip matrices," *Transportation Science*, vol. 20, no. 3, pp. 143–152, 1986.

[25] Y. Iida and J. Takayama, "Comparative study of model formulations on OD matrix estimation from observed link flows [C]," *Proceedings of 4th World Conference on Transportation Research*, vol. 2, pp. 1570–1581, 1986.

[26] E. Cascetta, "Estimation of trip matrices from traffic counts and survey data: a generalized least squares estimator," *Transportation Research Part B: Methodological*, vol. 18, no. 4-5, pp. 289–299, 1984.

[27] J. Xie, Y. Nie, and X. Liu, "A greedy path-based algorithm for traffic assignment," *Transportation Research Record*, vol. 2672, no. 48, pp. 36–44, 2018.

[28] S. D. Clark and D. P. Watling, "Applications of sensitivity analysis for probit stochastic network equilibrium," *European Journal of Operational Research*, vol. 175, no. 2, pp. 894–911, 2006.

[29] C. Yang and A. Chen, "Sensitivity analysis of the combined travel demand model with applications," *European Journal of Operational Research*, vol. 198, no. 3, pp. 909–921, 2009.

[30] M. Patriksson, "Sensitivity analysis of traffic equilibria," *Transportation Science*, vol. 38, no. 3, pp. 258–281, 2004.

[31] O. Drissi-Kaïtouni and J. T. Lundgren, *Bilevel Origin-Destination Matrix Estimation Using a Descent Approach*, Universitetet i Linköping/Tekniska Högskolan i Linköping. Department of Mathematics, Linköping, Sweden, 1992.

[32] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[33] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives," *Pacific Journal of Mathematics*, vol. 16, no. 1, pp. 1–3, 1966.

[34] Y. Nie, *A Programmer's Manual for Toolkit of Network Modeling (Tnm)*, University of California, Davis, CA, USA, 2006.

[35] D. Boyce, B. Ralevic-Dekic, and H. Bar-Gera, "Convergence of traffic assignments: how much is enough?" *Journal of Transportation Engineering*, vol. 130, no. 1, pp. 49–55, 2004.