Origin-Based Algorithms for Traffic Assignment

Algorithmic Structure, Complexity Analysis, and Convergence Performance

Jun Xie and Chi Xie

This paper presents an extensive analytical and numerical investigation of a class of origin-based algorithms for the user equilibrium–based traffic assignment problem. Nine known algorithms in this class are first clustered into four algorithmic structures on the basis of their structural differences and similarities in algorithm design. A complexity analysis of these algorithmic structures is conducted by calculating the frequency of executing node and link operations; this approach provides a simple analytical way to estimate their per iteration computation costs. To deliver a comprehensive and fair comparison of their convergence performance, all nine algorithms are implemented on the same programming platform and run to solve a few representative large-scale traffic networks by sizes and congestion levels. A close look at the convergence performance statistics further justifies the consistency of the complexity analysis and numerical evaluation results of the computational efficiency of these algorithms. Discussions on the degeneration of algorithm convergence efficiency with respect to network size and congestion level provide useful insights for the potential improvement of current origin-based algorithms or the proposition of new algorithms.

Traffic assignment has been widely used as the standard tool to predict network flow patterns in the final stage of travel demand forecasting. Although various traffic assignment models that can capture more realistic features of real traffic systems have been explored and studied, such as dynamic traffic assignment (*1, 2*) and stochastic traffic assignment models (*3, 4*), the most frequently used traffic assignment model in practice is still the static deterministic model under the user equilibrium (UE) principle, which states that for every origin–destination (O-D) pair all used routes have equal travel costs and no unused route has a lower cost (*5*). Several superiorities of the UE model are easy to enumerate, including its intuitive assumptions, simple formulations, and stable solutions with respect to inputs. To the authors' knowledge, the most important advantage of the UE model in comparison with other models may be attributed to its various kinds of solution methods that can efficiently solve large-scale network problems.

School of Naval Architecture and Ocean and Civil Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China. Corresponding author: C. Xie, chi.xie@sjtu.edu.cn.

Transportation Research Record: Journal of the Transportation Research Board, No. 2498, Transportation Research Board of the National Academies, Washington, D.C., 2015, pp. 46–55.
DOI: 10.3141/2498-06

The first class of practical traffic assignment problem (TAP) algorithms operates in the space of link flows. These algorithms are known as link-based algorithms; examples include the Frank–Wolfe algorithm and its variants (*6–8*). Easy implementation and fast convergence in early iterations have allowed these algorithms to emerge as the winner in the transportation planning practice during the past decades. However, the main shortcoming of the link-based algorithms is their inability to obtain high-precision solutions in most practical applications. To address that issue, path-based algorithms, which generate, store, and equilibrate a relatively small number of paths for each O-D pair, have been considered in the literature (*9–11*). In general, path-based algorithms work well for small- and medium-size problems, yet their efficiency degrades significantly with an increase in problem size (*12*).

This paper studies and compares a class of origin-based algorithms for the TAP that promise to produce highly precise UE solutions. All algorithms involved in this paper are characterized by the "origin-based" feature, by which the TAP is decomposed into a sequence of subproblems with respect to origins. Each one can be solved to an origin-based UE solution representation restricted on a single-origin network, and then these subproblems loop between all origins to equilibrate the entire network. Under the above relatively relaxed definition, a number of algorithms can be classified into this family, most of which were developed in the past decade.

The best known algorithm of this class is the origin-based algorithm (OBA) developed by Bar-Gera (*13*). Nie revised Bar-Gera's OBA by fixing an error in computing the second-order derivative and circumventing the line search in solving the restricted master problem (RMP) defined on each bush rooted at the origin (*14, 15*). In this paper, Nie's revised algorithm is called "QBA" following his original paper. Gentile proposed using the greedy method to solve the sequence of approximated quadratic programs and named the resulting algorithm the "local user cost equilibrium" (LUCE) algorithm (*16*). Dial proposed the so-called Algorithm B by solving the RMP directly by swapping flows from the longest path to the shortest path for each node on the bush, taking advantage of the acyclicity of a bush (*17*). Zheng and Peeta proposed a variant of Algorithm B by successively refining an endogenously determined ε value to solve the RMP into ε-optimal flows (*18*). The above five algorithms all recognize the fact that links with positive flows under UE will never form a directed cycle but will choose to maintain a bush for each origin and operate flows on bush links only. In that regard, these algorithms are also referred to as "bush-based" algorithms.

Another highly efficient algorithmic structure widely used in the design of origin-based algorithms is paired alternative segments

(PASs). The PAS concept was recently proposed by Bar-Gera although the use of it for the TAP actually has a much longer history (*19*). To the authors' best knowledge, the network simplex (NS) method adapted by Nguyen was the first algorithm that identifies and uses PASs for the TAP (*20*). It maintains a spanning tree for each origin such that a PAS can be detected by augmenting an out-of-tree link into the tree. This idea was recently revisited by Zheng, who fixed a pivot rule error of NS and proposed a generalization of this algorithm by allowing the spanning tree to contain links that could either emanate from or point toward the origin (*21*). For discussion convenience, this revised NS algorithm is abbreviated as RNS in this paper. Zheng's numerical results testified the high efficiency of using PASs in solving large-size problems. It is somewhat surprising that the NS algorithm has not received much attention although it has great potential to achieve high performance and was applied for the TAP earlier than the Frank–Wolfe algorithm. Another algorithm that uses PASs is Dial's Algorithm B, in which a bush is preserved for each origin instead (*17*). Thanks to the acyclicity of a bush, a PAS can be identified by following the shortest and longest path segments ending at the same node. The use of PASs for the TAP was more systematically explored by Bar-Gera in developing his famous traffic assignment by paired alternative segments (TAPAS) algorithm, in which a breadth first search method that can directly identify PASs from the shortest path tree rooted at each origin is applied (*19*). Xie and Xie recently improved TAPAS by proposing a new PAS identification method that continuously identifies the higher-cost segment with the maximum flow link in the backward set of the nodes and relates only one origin to each PAS (*22*). Their numerical experiments show that the improved TAPAS (called "iTAPAS") can be two times faster than the TAPAS in solving large networks. The above five algorithms all identify and operate PASs for a TAP formulated at one origin. Hence they can be called PAS-based algorithms. Algorithm B can be categorized as both a bush-based and a PAS-based algorithm as it uses bushes and PASs simultaneously.

In this paper the main research purpose is to make a fair and systematic comparison of the class of origin-based algorithms. For that purpose, all known origin-based algorithms and their variants were investigated and compared in the following two aspects. First, four algorithmic structures were provided to accommodate the nine algorithms of interest. Algorithms that use the same structure were compared by examining their similarities and differences in algorithmic details. For algorithms that belong to different structures, an attempt was made to compare the computation costs of each iteration by estimating the frequency of scanning the set of nodes and links for each structure. (Hereafter in this paper, such an evaluation of the computation costs pertaining to these different algorithmic structures is called "structural complexity," which is different from the widely used definition of computational complexity. The latter typically takes into account very basic algorithmic operations, such as assignment and arithmetic and logical operations, in the worst case of problem instances.) Second, all nine algorithms were implemented on the same programming platform, and different algorithms were allowed to share as many subroutines as possible. Extensive numerical experiments were conducted on both medium- and large-scale test problems to provide an overall comparison of convergence efficiency. Several per iteration convergence statistics are examined, and insights and explanations for the convergence behavior of different algorithms are provided.

Extensive and systematic comparisons of the latest UE algorithms are important and essential to transportation planning practitioners and software developers since they need to know which algorithms will satisfy their needs. Inoue and Maruyama conducted a very good comparison of the traditional link-based and path-based algorithms and some of the origin-based algorithms (*12*). Their work is advanced here by the addition of several new algorithms and by focusing only on the class of state-of-the-art origin-based algorithms. A contribution to the literature through this paper is also made by an estimation of the structural complexity of these algorithms, which provides an analytical way to assess and compare the computation costs of the algorithms.

The rest of this paper is organized as follows. The next section describes the notation and algorithmic terms that are used in this paper. The nine algorithms considered are then briefly reviewed, and a comparison is done by categorizing them into four algorithmic structures and estimating their structural complexity. Next the convergence performance of the nine algorithms is compared and analyzed. The paper concludes with a few remarks on the major findings and future research.

NOTATION AND ALGORITHMIC TERMS

Consider a directed traffic network $G(N, A)$ where N and A denote the sets of nodes and links, respectively. Suppose G is strongly connected, that is, there is at least one path connecting any two nodes in G. N consists of a set of nodes; some are origins denoted by R and some are destinations denoted by S. The travel demand departing from an origin $r \in R$ to a destination $s \in S$ is denoted by d_{rs}. Link (i, j) is directed from node i to node j. Let x_{ij} be the traffic flow on link $(i, j) \in A$ and $t_{ij}(\cdot)$ be the travel time of traversing link (i, j), which is a strictly positive, increasing, and convex function of x_{ij}.

Given the above network, the UE-based TAP can be formulated as a convex optimization problem with linear constraints. Existing algorithms usually decompose a TAP into a series of subproblems and solve them iteratively. Central to the development of origin-based algorithms is the decomposition of traffic assignment with respect to origins (or destinations). More specifically, a TAP is decomposed into a sequence of single-origin TAPs, each one being defined on a subnetwork. Definitions follow:

Definition 1. Subnetwork. A "subnetwork" is a duplicate of network G but is rooted at a single origin r. Only the demand departing from origin r can be assigned to links of the subnetwork. Let $G_r(N, A_r)$ denote a subnetwork.

Maintaining a subnetwork for each origin is essential for origin-based algorithms. Link flows need to be disassembled into the origin-specific link level and to be stored on the links of the subnetwork, denoted by x^r_{ij}. Origin-based link flows on the subnetworks contain much detailed information that can be used to generate precise descent directions. Subnetworks usually provide only a platform to decompose a TAP into multiple single-origin TAPs. They do not have the potential to force a feasible flow pattern to be optimal. Next, several other algorithmic devices that provide key blocks of methods used for solving these single-origin TAPs are given below.

Definition 2. Bush. A "subnetwork" is called a bush if it is acyclic. It is denoted by $B(N, A_B)$. Some network operations (such as a shortest-path search) become more efficient in a bush by making use of topological distance, which is defined next.

Definition 3. Topological distance. Let the length of every link in a bush be 1. The topological distance of a node j, denoted as π_j, is the maximum distance from the origin to node j (*14*).

Definition 4. Ascending (descending) pass. An "ascending" ("descending") pass is a sequential visit to each node of a bush following the increasing (decreasing) order of topological distance (*14*). On a bush, the shortest-path and the longest-path trees can be built quickly and simultaneously by scanning each node and link once.

Definition 5. Spanning tree. A "bush" is a spanning tree if it contains a minimal set of links that connect all nodes.

Definition 6. Segment. A "segment" is defined as a sequence of distinct nodes; there is only one directed link between every pair of nodes. The beginning and ending nodes of a segment are called the "tail" and "head," respectively. A segment is called a "directed segment" if links of the segment are all directed from the tail to the head.

Definition 7. Cycle. A segment and one link that connects the tail and the head of this segment constitute a cycle. A cycle is called a "directed cycle" if all links of the cycle are directed clockwise or anticlockwise.

Definition 8. Paired alternative segments. A "set of PASs" is defined as two segments that have the following features: (*a*) they have the same tail and head nodes and there are no more common nodes and (*b*) they are both directed segments (*19*).

OVERVIEW AND COMPLEXITY ANALYSIS

In this section an overview of the origin-based algorithms and their algorithmic structures is provided. Nine algorithms are categorized into four algorithmic structures according to their algorithmic design resemblance. On the basis of these structures, the key differences and similarities of the algorithms in each algorithmic structure are first reviewed; then the algorithms are compared across different implementation structures by approximating the structural complexity of their subproblems, which can be used to assess computing costs in individual iterations.

Algorithm Overview

Algorithmic Structure 1

OBA, QBA, and LUCE share the same algorithmic structure as described in the following. In Step 1, the bush management strategies suggested by Nie are used for all three algorithms (*14*). The key difference between them is in Step 2.2, in which the RMP is decomposed into a sequence of quadratic approximation problems defined on each node; these problems are solved subsequently to provide a descent direction. OBA uses a gradient projection algorithm for these quadratic approximation problems; LUCE instead uses a greedy algorithm. QBA follows OBA by using the gradient projection algorithm for quadratic approximation problems, but it improves the veracity of these approximation problems by updating the cost and derivatives of links in the upper stream after each quadratic approximation problem is solved. OBA and LUCE need to adjust the step size of the line search for the RMP. OBA invokes a descending pass for each line search adjustment to make sure that the descent direction obtained can always diminish the objective function. LUCE does not invoke a descending pass, and QBA further waives the line search.

Step 0. Initialization. For each origin, initialize the bush as a shortest-path tree rooted at the origin. Assign all flows to links on the tree.

Step 1. Bush management. For each bush, do the following:

1.1. Update the shortest-path tree and node labels by an ascending pass, and scan each link in the bush once to trim the bush links.

1.2. Update the topology order, and then update the shortest- and longest-path tree and node labels; scan every link out of the bush to expand the bush. Update the topology order again.

Step 2. Bush equilibration. Repeat the following steps 20 times at most. For each bush, do the following:

2.1. Calculate the shortest-path tree; update node and link labels by an ascending pass.

2.2. Solve the approximated quadratic problem defined on each node by a descending pass.

Step 3. Convergence check. If the convergence criterion is achieved, stop; otherwise, go to Step 1.

Algorithmic Structure 2

Algorithm B and εBA share Algorithmic Structure 2. In the present implementation, the bush management strategy adopted in Structure 2 is the same as the strategy in Structure 1; their main difference lies in the bush equilibration step. In Step 2, Algorithm B calculates the longest- and shortest-path cost from the origin to each node in the bush; if the cost difference is larger than an exogenously given precision ε, then a PAS is searched and equilibrated for this node. Here the choice of the value of ε appears to be a concern for Algorithm B in the sense that if a relatively large value of ε is chosen, the algorithm may then fail to converge to an expected precision. If a smaller value is chosen, then the number of PASs identified in each iteration may be too large to be needed, that is, in the early stage of algorithms when the relative gap is still much larger than ε, identifying PASs with a relatively small cost difference usually has negligible effects on convergence but would produce considerable computational effort. εBA proposes to successively refine an endogenously determined ε value to control the generation of PAS in each iteration more elaborately according to convergence needs. In εBA, ε is initialized as a relatively large value and refined in each iteration, and it reduces to zero gradually as the algorithm converges to the UE solution. One possible shortcoming of εBA is that one has to choose an initial value of ε and set a proper refinement process for it in the algorithmic procedure, mostly according to experience.

Step 0. Initialization. For each origin, initialize the bush as a shortest-path tree rooted at the origin. Assign all flows to links on the tree.

Step 1. Bush management. For each bush, do the following:

1.1. Update the shortest-path tree and node labels by an ascending pass; scan each link in the bush once to trim the bush links.

1.2. Update the topology order, and then update the shortest- and longest-path tree and node labels; scan every link out of the bush to expand the bush. Update the topology order again.

Step 2. Bush equilibration. Repeat the following steps 20 times at most. For each bush, do the following:

2.1. Calculate the shortest-path and longest-path trees; update the node labels by an ascending pass.

2.2. Perform a descending pass to search and equilibrate a PAS for each node so that its longest-path and shortest-path cost difference is larger than a specific coefficient ε. One segment of the PAS is searched following the shortest path, and the other follows the longest path.

Step 3. Convergence check. If the convergence precision is achieved, stop; otherwise, go to Step 1.

Algorithmic Structure 3

NS and RNS share Algorithm Structure 3. Other than a subnetwork, NS maintains a directed spanning tree for each origin to identify a PAS by adding an out-of-tree link (*i, j*) to the tree. The identified PAS is then equilibrated by shifting flows from the higher-cost segment to the lower-cost segment to force the reduced cost of link (*i, j*) to zero. If one link on the higher-cost segment has zero origin flow, then the PAS will be unable to shift any amount of flow for link (*i, j*) for this origin. A search of this kind is called a "degenerated search." Degenerated searches may incur useless computation efforts or even make the algorithm unable to converge. For this reason, special techniques are usually needed for NS to handle degenerated searches. After the PAS is equilibrated, the tree is updated by removing the zero-flow link from the tree and adding a nonzero link to keep the tree connected. Zheng fixed an error in the pivot rule of NS that may generate directed cycles and subsequently fail to update the tree properly (*21*). He further proposed a variant of this algorithm (named "RNS") that allows the spanning tree to contain links that could be directed either away from or toward the origin, such that his revised version of NS searches cycles instead of PASs. The superiority of such a revision is that a strongly feasible spanning tree can be maintained for each origin, which will not generate degenerated searches; the shortcoming is that it is more complex to implement the algorithm.

Step 0. Initialization. For each origin, initialize the subnetwork and the spanning tree as the shortest-path tree. Assign all flows to links on the tree.

Step 1. For each spanning tree, calculate the path cost from the origin to each node by a depth first algorithm.

Step 2. Repeat the following steps 20 times at most. For each origin, do the following:

2.1. Scan every link out of the spanning tree; if the reduced cost of link (*i, j*) is negative or positive but with nonzero origin flow, search a PAS or cycle in Step 2.2.

2.2. Search a PAS or a cycle by adding link (*i, j*) into the spanning tree, and augment flows around the PAS or cycle to equilibrate the cost. Update the spanning tree with a pivot rule.

Step 3. Convergence check. If the convergence precision is achieved, stop; otherwise, go to Step 1.

Algorithmic Structure 4

TAPAS and iTAPAS share Algorithmic Structure 4. TAPAS systematically explores the use of PAS for solving the TAP, and its three notable innovations are summarized as follows: (*a*) it uses a breadth first search method to identify PASs directly from the shortest-path tree rooted at each origin at Step 1.2; (*b*) it stores PASs in a list to save the time spent on repeatedly identifying the same segments at Step 1.2; and (*c*) it connects multiple origins to one PAS by matching an origin to an existing PAS in the list, a process that helps secure a unique path solution by forcing proportionality. Xie and Xie examined the TAPAS algorithm in great detail and found several issues related to the identification and operations of PASs that may affect the overall algorithm convergence efficiency, including (*a*) the breadth first search method chooses the higher cost segment arbitrarily, which could lead to unpredictable and inconsistent behavior; (*b*) the breadth first search method may fail to identify a PAS successfully if a flow effective factor is imposed, such that the algorithm convergence cannot be guaranteed unless an additional branch shift operation is conducted; and (*c*) the association of multiple origins with one PAS may incur a large number of link flow operations that lag the algorithm convergence in solving large, congested networks. The resulting improved algorithm, "iTAPAS," differs from TAPAS mainly in the following aspects: (*a*) a new PAS identification method that is easier, faster, and more effective is proposed instead of the breadth first search method and the branch shift operation; the new method identifies the higher cost segment continuously with the maximum flow link in the backward set of the node that starts from the tail node of the target link to the origin; and (*b*) only one origin is related to each PAS in Step 1.2 in the algorithm convergence process, and postprocessing can be added to achieve the proportionality. Implementation of iTAPAS becomes significantly easier than that of TAPAS (*22*).

Step 0. Initialization. For each origin, initialize the subnetwork as the shortest-path tree. Assign all flows to links on the tree.

Step 1. For each origin, do the following:

1.1. Calculate the shortest-path tree rooted at the origin.

1.2. Scan each link out of the tree. If the origin link flow is larger than 0 and the reduced cost is smaller than 0, match an existing PAS from the PAS list for this link; if an existing PAS is not matched, identify a new PAS according to the shortest-path tree. Equilibrate the segments cost once for the new identified PAS, and save it into the PAS list if both segments still have positive flows.

1.3. Select a number of PASs (say, 400) from the PAS list, and equilibrate the segments cost once.

Step 2. Repeat the following step 20 times at most: scan each PAS in the list; if the flow (contributed by all associated origins) on one segment reduces to 0, delete it from the PAS list; otherwise, equilibrate the segments cost once for this PAS.

Complexity Analysis

Given the above structures, the algorithms accommodated by the same structure tend to share more common features than those in different structures. Here an evaluation will be done of the structural complexity of different algorithmic structures in each iteration by estimating the frequency that they scan the set of nodes and links, which is usually the most expensive step in TAP algorithms. Such a complexity analysis provides one with helpful insights on how much computation effort is needed to conduct one algorithmic iteration by different algorithmic structures. Table 1 reports the structural complexity of different algorithmic structures. Scanning a general network is commonly more expensive than scanning a bush or a tree. Here they are not distinguished because the scanning

TABLE 1 Structural Complexity of Algorithmic Structures

Structure	Algorithm	Complexity
1	OBA, QBA, LUCE	$46R$
2	B, εBA	$26R$
3	NS, RNS	$21R$
4	TAPAS, iTAPAS	$2R + 20P$

NOTE: R = number of origins; P = PAS list size.

process involves link flow and cost operations that usually spread over the whole node set and link set, so the structural complexity difference caused by different network topology is negligible. Let R denote the number of origins and P denote the number of PASs. Structure 1 and Structure 2 share the same bush management strategies, so the structural complexity of Step 1 for the two structures is $6R$. The structural complexity of the bush equilibration (Step 2) in Structure 1 is roughly estimated as $40R$ and that in Structure 2 as $20R$. A fast implementation strategy suggested by Zheng and Peeta for B and εBA is used, such that more than 50% of the network scanning usually can be omitted in Structure 2 (*18*). Structure 3 scans the network only once for each origin in Step 2 such that its structural complexity is estimated as $20R$. The structure of TAPAS and iTAPAS has one major difference from the other three structures in that it stores PASs into a list in Step 1 and subsequently obviates scanning over the network in Step 2. When the number of PASs in the list is smaller than the node set size, Structure 4 spends less computational effort than that of Structure 3; as the PAS list size increases to a size that is comparable with that of nodes or links, then the computation effort of Structure 4 is roughly equal to or even larger than that of Structure 3.

The structural complexity estimation cannot reflect the algorithm convergent efficiency unless one further knows the average convergence magnitude achieved by each iteration. Actually, these two parts are subject to a trade-off in designing a TAP algorithm. Take Algorithm B described in Structure 2 as an example. A larger number of repetitions of bush equilibration will invoke more ascending and descending passes of the bush and thus increase the computation time of each iteration, but it also will let the RMP more converged. Even so, the complexity analysis is still very useful in explaining the present numerical experiment results and making the comparison results more convincible in the next section, which discusses the average computation times and the average convergence magnitude per iteration being calculated on the basis of the assignment results.

NUMERICAL ANALYSIS

The nine origin-based algorithms will be compared further by conducting a set of numerical experiments. This section is organized into three parts. The first describes the algorithm implementation environment and the test networks used in the numerical experiments. The performance of the nine algorithms in solving different networks are compared in the second part. The last part examines the average computation time and the average convergence magnitude by each iteration of the algorithms for large networks.

Computing Environment and Test Networks

All nine algorithms are coded by using the toolkit of network modeling, a C++ class library specialized in modeling transportation networks (*23*). In the present implementation, different algorithms share as many subroutines as they can; their main differences are all described according to the four algorithmic structures identified in the previous section. All numerical results reported in this section were produced on a Windows 8 64-bit workstation with Core i5 3.2 GHz CPU and 16 G random-access memory.

The algorithms are applied to six networks; two networks are medium size and four networks are large scale. The profile information of these networks are reported in Table 2. The last column shows the average volume-to-capacity (V/C) ratio of networks at user equilibrium; one can see that Austin is the most congested network and PRISM is the least congested in the four large regional networks. The main convergence indicator used in this study is the relative gap, which measures how close the current solution is to the true user equilibrium. In the present experiments, the relative gap (RG) is calculated as

$$\mathrm{RG} = 1 - \frac{\sum_{rs} u_{rs} q_{rs}}{\sum_{(i,j)\in A} x_{ij} t_{ij}} \tag{1}$$

where u_{rs} is the minimum travel cost between O-D pair r-s according to the current link travel cost t_{ij}, and q_{rs} is the travel demand between the O-D pair r-s. Finally, the Bureau of Public Roads function is used to calculate link travel costs.

Overall Convergence Performance

This section compares the convergence performance of the nine algorithms on different networks. The convergence criterion is set to 10^{-1}, and the maximum running time is set to 6 h for all experiments. Figure 1 illustrates the evolution of different algorithms in solving the six networks. The horizontal axis of these graphs denotes the computation times in seconds/minutes, the vertical axis denotes the RG in the logarithmic scale. All nine algorithms can solve the two medium-size networks to a relative gap of 10^{-12} within 2 min, and hence testify their capacity of achieving highly precise solutions. The convergence capacity of all algorithms tends to degrade with the size and congestion level of the test networks. For the four larger regional networks, most algorithms except for LUCE can converge

TABLE 2 Profile Information of Test Networks Used in Numerical Experiments

Network	Number of Nodes	Number of Links	Number of Zones	Number of O-D Pairs	Average Link v/c Ratio
Medium					
Winnipeg	1,050	2,836	135	4,345	0.33
Chicago Sketch	933	2,950	387	142,890	0.54
Large					
PRISM	14,639	33,937	898	453,089	0.26
Chicago regional	12,982	39,018	1,771	3,136,441	0.60
Philadelphia	13,389	40,003	1,525	1,150,722	0.75
Austin	7,388	18,956	1,117	1,080,695	1.14

NOTE: v/c = volume to capacity.

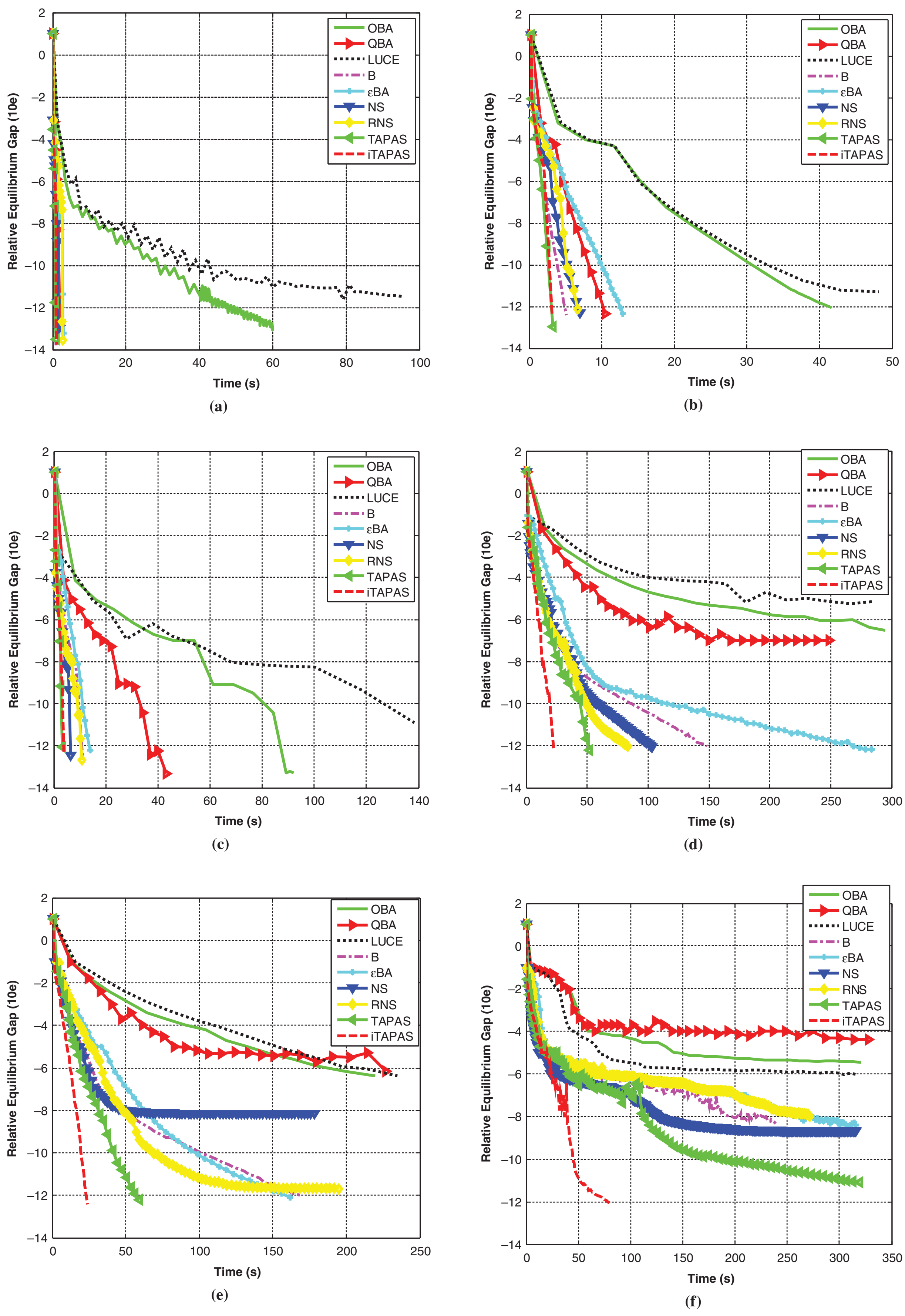


FIGURE 1 Convergence performance of nine algorithms on test networks: (*a*) Winnipeg, Manitoba, Canada; (*b*) Chicago, Illinois, Sketch; (*c*) PRISM; (*d*) Chicago regional; (*e*) Philadelphia, Pennsylvania; and (*f*) Austin, Texas.

to an RG of 10^{-12} in solving the less congested PRISM network, and nearly all algorithms other than iTAPAS fail to reach an RG of 10^{-1} in solving the most congested Austin network. Apparently, how much the performance of an algorithm degrades with increasing size and congestion level is a critical consideration for researchers and practitioners in choosing an algorithm, and TAPAS or iTAPAS is usually preferable to OBA or LUCE.

Particularly, iTAPAS is found to be the algorithm with the best performance in all cases, and TAPAS follows as the second-best algorithm. iTAPAS can be two times faster than TAPAS in solving the Chicago regional, Philadelphia, and Austin, Texas, networks, and this finding may be attributed to the new PAS identification method and the simplified PAS flow operations of iTAPAS. NS, RNS, Algorithm B, and εBA perform similarly in the present test. RNS may outperform the other three slightly if examined closely. NS stops to converge further at an RG of 10^{-8} in solving the Philadelphia and Austin networks, for unclear reasons. OBA, QBA, and LUCE perform less efficiently than the other six algorithms to a large extent, owing mainly to their unfaithful approximation of RMP into a sequence of quadratic node problems. For detailed discussions on these three algorithms see Xie et al. (*24*).

Convergence Performance per Iteration

In this section, the assignment results will be further explored, particularly in several per iteration statistics, to provide more comparisons in the convergence behavior of different algorithms and to give possible explanations for the degradation effects of algorithms with network size and congestion level. Recall that the overall convergence of algorithms can be determined by two aspects: the average computation time per iteration and the average convergence magnitude per iteration. Given the assignments results, one can easily calculate the average computation time of one iteration for each algorithm on a specific network. Here an indicator *M* is further defined to measure the average convergence magnitude of each iteration as follows:

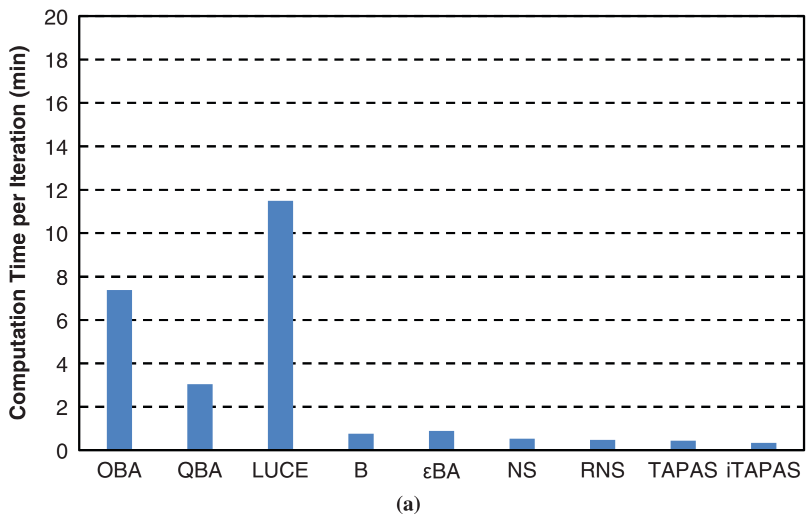
$$M = \frac{\sum_{k\in[1\ldots K-1]} \left(\log \mathrm{RG}_k - \log \mathrm{RG}_{k+1}\right)}{K-1} \qquad (2)$$

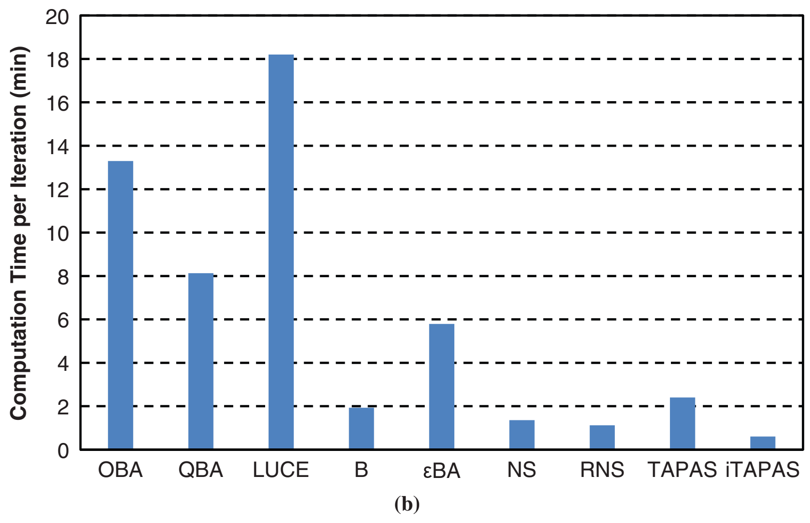
where RG_k denotes the relative gap defined in Equation 1 on iteration *k*, and *K* is the maximum iteration index when the algorithms converge to an RG of 10^{-12} or reach the maximum running time.

The two computation statistics averaged by all iterations for each algorithm and each network are reported in Table 3. *K* denotes the total number of iterations considered, *T* denotes the average computation time of one iteration, and *M* is the convergence magnitude indicator defined above. Some useful conclusions can be obtained from this table. By taking the statistics of LUCE in solving the Chicago regional network as an example, it was found that it takes about 18.2 min for LUCE to achieve an *M* of .23. If it is optimistically assumed that *T* and *M* in this case will remain unchanged through the whole convergence process, then it is easy to calculate that LUCE will take at least 15.8 h to converge to an RG of 10^{-12} in this case. Earlier in the paper, an estimation of the structural complexity of each structure was proposed to obtain a glance at the computation effort and time needed for one iteration. The real computation times shown in Table 3 show a consistent pattern with the prediction of the structural complexity. These computation times are plotted in Figure 2. The per iteration computation times of OBA, QBA, and LUCE are clearly larger than those of the other six algorithms in all cases, which is consistent with the complexity analysis results. iTAPAS is always the algorithm with the lowest computation time in all test cases. The remaining five algorithms are comparable, and a close look reveals that the computation times of Algorithm B and εBA are slightly larger than those of NS and RNS. The computation time of TAPAS seems to heavily depend on the number of PASs maintained. The present assignment results report that the average number of PASs maintained by TAPAS per iteration for the PRISM, Chicago regional, Philadelphia, and Austin networks is 333, 14,792, 48,913 and 81,768, respectively. As the number of PASs increases, the computation time required per iteration increases accordingly. The variations in computation times of iTAPAS are less obvious for the following two reasons: (*a*) iTAPAS maintains fewer PASs than TAPAS does per iteration owing to the more effective PAS identification; those numbers are 253, 9,692, 20,017, and 56,429 for the PRISM, Chicago regional, Philadelphia, and Austin networks, respectively; and (*b*) iTAPAS associates only one origin with each PAS, so its operations on a PAS take less time than that of TAPAS, which relates multiple origins to one PAS.

TABLE 3 Average Computation Time and Convergence Magnitude per Iteration

		Algorithm								
Network	Statistics per Iteration	OBA	QBA	LUCE	B	εBA	NS	RNS	TAPAS	iTAPAS
PRISM	*K*	12	14	12	13	15	11	22	6	10
	T (min)	7.38	3.04	11.5	.76	.89	.53	.48	.44	.34
	M	.816	.902	.695	.776	.644	.881	.472	1.90	1.26
Chicago regional	*K*	22	29	12	76	51	75	73	21	36
	T (min)	13.3	8.13	18.2	1.93	5.79	1.36	1.12	2.4	.60
	M	.240	.182	.23	.144	.217	.141	.145	.503	.295
Philadelphia	*K*	18	34	20	54	43	94	79	23	26
	T (min)	16.5	10.26	17.3	3.13	3.74	1.88	2.43	2.45	.86
	M	.342	.165	.314	.204	.258	.076	.135	.467	.421
Austin	*K*	50	44	74	199	145	119	199	95	133
	T (min)	7.25	8.37	4.83	1.20	2.1	.95	1.36	3.4	.67
	M	.089	.091	.07	.037	.05	.044	.034	.1	.091





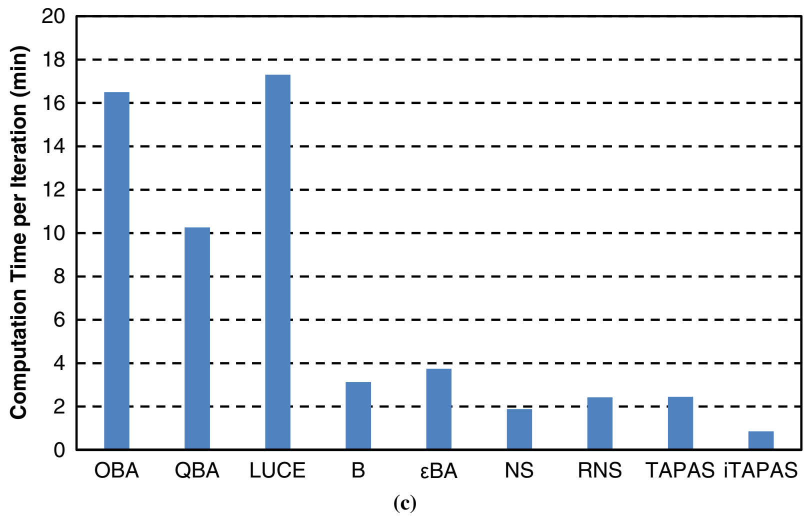


FIGURE 2 Average computation time per iteration: (*a*) PRISM, (*b*) Chicago regional, and (*c*) Philadelphia.

(continued on next page)

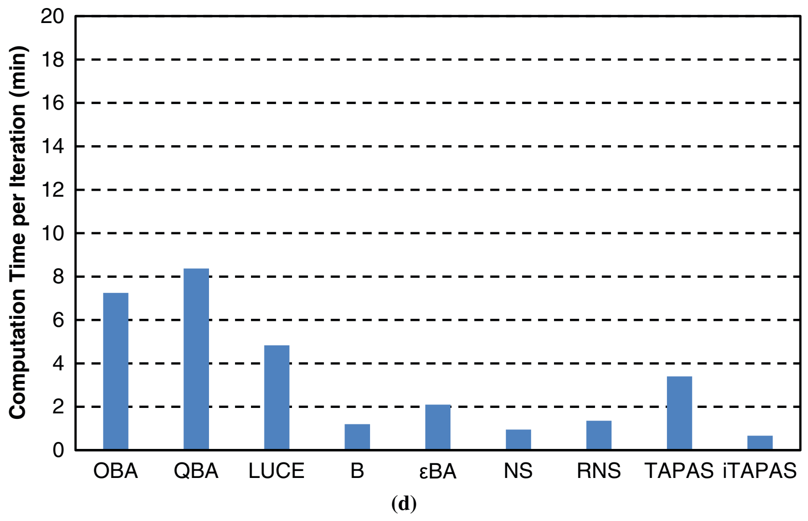


FIGURE 2 *(continued)* Average computation time per iteration: (*d*) Austin.

On the basis of the iteration statistics in Table 3, one can further calculate the algorithm convergence rate (CR) as follows:

$$\text{CR} = \frac{M}{T} \tag{3}$$

The above convergence rate can reflect the average convergence magnitude in relation to time. Figure 3 shows the convergence rate of different algorithms for the Chicago regional network. It reveals that iTAPAS is the most time-effective algorithm in this example, and it is more than two times faster than TAPAS.

CONCLUDING REMARKS

In this paper, the class of origin-based algorithms for the TAP under the UE principle is investigated and compared in both the analytical and the numerical aspects. All nine known origin-based algorithms are accommodated into four algorithmic structures, each of which is distinguished from the others in regard to some structural differences. These algorithms were implemented on the same programming platform, which allows one to conduct a fair comparison of their convergence performance. The results of numerical experiments reveal that the relative performance of the algorithms is consistent with the complexity analysis reported in this paper.

All the origin-based algorithms have the capacity of achieving a highly precise solution for the TAP under the UE principle. However, the numerical results here show that the level of precision that each algorithm is capable of achieving tends to degrade with increasing network size and congestion level. It seems that the worse an algorithm performs, the more it degrades. This phenomenon was further explored by decomposing the convergence efficiency into two parts: the average computation time and the average convergence magnitude per iteration. The algorithms with a worse performance usually have a larger per iteration computation time because they scan the network more frequently and have a smaller per iteration convergence magnitude because they usually execute link flow and cost operations more ineffectively.

Hence, one can, of course, improve the efficiency of the origin-based algorithms by either reducing the computation time per iteration

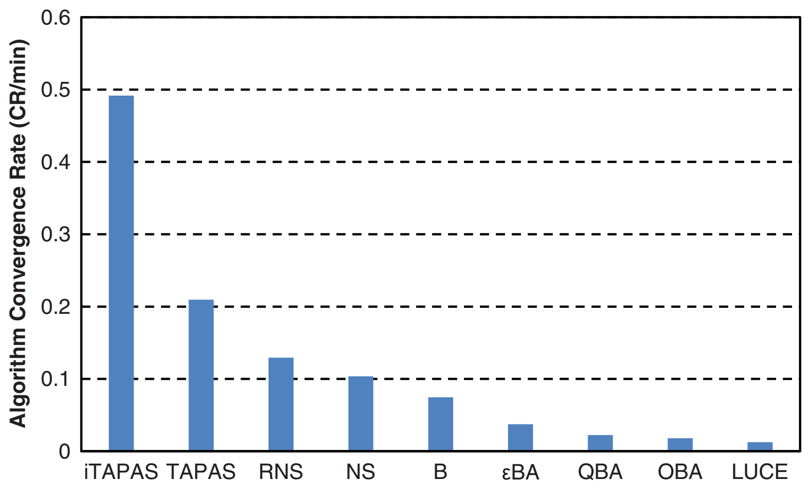


FIGURE 3 Convergence rate of algorithms for Chicago regional network.

or increasing the convergence magnitude per iteration. However, these two aspects usually are subject to a trade-off. The complexity analysis and numerical computation results show that the average computation time per iteration is closely relevant to the frequency of network scanning. Many existing algorithms were designed to scan the origin list (or the subnetwork list) multiple times in each main iteration to increase the convergence magnitude per iteration, at the expense of increasing the computation time per iteration. This technique is quite effective in fastening the overall convergence efficiency when the network size is small or medium and the computation time incurred by repeating network scans is not considerable. In the case of large regional networks, Nie suggested checking a convergence criterion before an origin is scanned, such that the scanning operation on this subnetwork can be skipped if it is well converged (*14*). Zheng further proposed to skip *K* times of scanning operations if one subnetwork is converged under a certain precision, such that the computation time per iteration diminishes significantly while the convergence magnitude remains unchanged (*21*). Following that line, TAPAS expends its greatest computation effort on the flow and cost operations of PASs while successfully avoiding repeated scans of subnetworks, an approach that partly accounts for its high efficiency (*19*). The analytical and numerical analysis in this paper well verifies the above experience in a full-scale perspective.

The past decade observed the development and improvement of various origin-based algorithms for TAPs of the UE type. The limited experience suggests that the following aspects may deserve further research efforts in the future. First, in this paper, the main focus was on an implementation and numerical comparison. However, the theoretical and technical reasons behind the algorithm performance are rarely discussed. Hence, a comprehensive review of existing mathematical and algorithmic principles on traffic assignment algorithms is valuable and desired. Second, given that most efficient algorithms for large-scale problems presented in this paper use the PAS structure, a further exploration and application of PAS for traffic assignment algorithms or for algorithms for other types of network equilibrium problems are recommended. Third, the use of PAS and many other algorithmic devices in origin-based algorithms naturally allows the implementation of parallel computing. Introducing parallel computing techniques into origin-based algorithms is another exciting research direction.

ACKNOWLEDGMENTS

The authors are grateful to Yu Nie of Northwestern University for his kindness in sharing the toolkit from the network modeling programming package, without which this work could not be completed. The numerical results of OBA, QBA, and LUCE were produced by the code written by Yu Nie, and the code for the rest of the algorithms was written by the first author. The authors also thank the four anonymous reviewers for their valuable comments. This work is sponsored jointly by a research project awarded to the first author by the China Postdoctoral Science Foundation and a research grant awarded to the second author from the State Key Laboratory of Ocean Engineering at Shanghai Jiao Tong University.

REFERENCES

1. Merchant, D.K., and G.L. Nemhauser. Model and an Algorithm for the Dynamic Traffic Assignment Problems. *Transportation Science,* Vol. 12, No. 3, 1978, pp. 183–199.
2. Peeta, S., and A.K. Ziliaskopoulos. Foundations of Dynamic Traffic Assignment: The Past, the Present, and the Future. *Networks and Spatial Economics,* Vol. 1, No. 3-4, 2001, pp. 233–266.
3. Daganzo, C.F., and Y. Sheffi. On Stochastic Models of Traffic Assignment. *Transportation Science,* Vol. 11, No. 3, 1977, pp. 83–111.
4. Xie, C., and S.T. Waller. Stochastic Traffic Assignment, Lagrangian Dual, and Unconstrained Convex Optimization. *Transportation Research Part B: Methodological,* Vol. 46, No. 8, 2012, pp. 1023–1042.
5. Wardrop, J.G. Some Theoretical Aspects of Road Traffic Research. *Proceedings of the Institution of Civil Engineering, Part II,* Vol. 1, No. 2, 1952, pp. 325–378.
6. LeBlanc, L.J., E.K. Morlok, and W.P. Pierskalla. An Efficient Approach to Solving the Road Network Equilibrium Traffic Assignment Problem. *Transportation Research,* Vol. 9, No. 5, 1975, pp. 309–318.
7. Florian, M., J. Guálat, and H. Spiess. An Efficient Implementation of the "Partan" Variant of the Linear Approximation Method for the Network Equilibrium Problem. *Networks,* Vol. 17, No. 3, 1987, pp. 319–339.
8. Fukushima, M. A Modified Frank–Wolfe Algorithm for Solving the Traffic Assignment Problem. *Transportation Research Part B: Methodological,* Vol. 18, No. 2, 1984, pp. 169–177.
9. Larsson, T., and M. Patriksson. Simplicial Decomposition with Disaggregated Representation for the Traffic Assignment Problem. *Transportation Science,* Vol. 26, No. 1, 1992, pp. 4–17.
10. Jayakrishnan, R., W.K. Tsai, J.N. Prashker, and S. Rajadhyaksha. Faster Path-Based Algorithm for Traffic Assignment. In *Transportation Research Record 1443,* TRB, National Research Council, Washington, D.C., 1994, pp. 75–83.
11. Florian, M., I. Constantin, and D. Florian. A New Look at Projected Gradient Method for Equilibrium Assignment. In *Transportation Research Record: Journal of the Transportation Research Board, No. 2090,* Transportation Research Board of the National Academies, Washington, D.C., 2009, pp. 10–16.
12. Inoue, S., and T. Maruyama. Computational Experience on Advanced Algorithms for User Equilibrium Traffic Assignment Problem and Its Convergence Error. *The 8th International Conference on Traffic and Transportation Studies,* Vol. 43, Changsha, China, 2012, pp. 445–456.
13. Bar-Gera, H. Origin-Based Algorithm for the Traffic Assignment Problem. *Transportation Science,* Vol. 36, No. 4, 2002, pp. 398–417.
14. Nie, Y.M. A Class of Bush-Based Algorithms for the Traffic Assignment Problem. *Transportation Research Part B: Methodological,* Vol. 44, No. 1, 2010, pp. 73–89.
15. Nie, Y.M. A Note on Bar-Gera's Algorithm for the Origin-Based Traffic Assignment Problem. *Transportation Science,* Vol. 46, No. 1, 2012, pp. 27–38.
16. Gentile, G. Local User Cost Equilibrium: A Bush-Based Algorithm for Traffic Assignment. *Transportmetrica,* 2012.
17. Dial, R.B. A Path-Based User-Equilibrium Traffic Assignment Algorithm That Obviates Path Storage and Enumeration. *Transportation Research Part B: Methodological,* Vol. 40, No. 10, 2006, pp. 917–936.
18. Zheng, H., and S. Peeta. Cost Scaling Based Successive Approximation Algorithm for the Traffic Assignment Problem. *Transportation Research Part B: Methodological,* Vol. 68, 2014, pp. 17–30.
19. Bar-Gera, H. Traffic Assignment by Paired Alternative Segments. *Transportation Research Part B: Methodological,* Vol. 44, No. 8, 2010, pp. 1022–1046.
20. Nguyen, S. An Algorithm for the Traffic Assignment Problem. *Transportation Science,* Vol. 8, No. 3, 1974, pp. 203–216.
21. Zheng, H. Adaptation of Network Simplex for the Traffic Assignment Problem. *Transportation Science,* 2015 (forthcoming).
22. Xie, J., and C. Xie. An Improved TAPAS Algorithm for the Traffic Assignment Problem. *Proc., 17th International IEEE Conference on Intelligent Transportation.* Qingdao, China, 2014.
23. Nie, Y. *A Programmer's Manual for Toolkit of Network Modeling* (*TNM*). Department of Civil and Environmental Engineering, University of California, Davis, 2006.
24. Xie, J., Y.M. Nie, and X. Yang. Quadratic Approximation and Convergence of Some Bush-Based Algorithms for the Traffic Assignment Problem. *Transportation Research Part B: Methodological,* Vol. 56, 2013, pp. 15–30.

Any errors are the responsibility of the authors.

The Standing Committee on Transportation Network Modeling peer-reviewed this paper.