Transportation Research Part B 93 (2016) 406–424

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

New insights and improvements of using paired alternative segments for traffic assignment

Jun Xie[a,1], Chi Xie[a,b,*]

[a] School of Naval Architecture, Ocean and Civil Engineering, Shanghai Jiaotong University, China
[b] State Key Laboratory of Ocean Engineering, Shanghai Jiaotong University, China

ARTICLE INFO

Article history:
Received 10 July 2015
Revised 21 August 2016
Accepted 22 August 2016

Keywords:
Traffic assignment
User equilibrium
Paired alternative segments
Consistency and proportionality

ABSTRACT

The recent literature observes that the development of advanced algorithms for the traffic assignment problem (TAP) heavily relies on the proper use of some specific topological structures. This paper focuses on discussing a particular topological structure named paired alternative segment (PAS), which consists of two path segments sharing the same starting and ending nodes but no other common nodes. We first present two alternative conditions that establish an equivalency relationship between user equilibrium (UE) flows and PAS structures. Starting from the traffic assignment method by paired alternative segments (TAPAS), we then examine the utilization of PASs for TAP and explored some algorithmic and implementation issues, which leads to the birth of an improved TAPAS procedure (termed iTAPAS in this paper). Compared to the original TAPAS, iTAPAS enhances the algorithmic efficiency in two aspects: (1) a more effective PAS identification method is used; (2) each PAS is set as being associated with only one origin in the UE-finding process. Some analytical results based on the new PAS identification method are presented to justify the convergence and efficiency of iTAPAS. A simplified post-process procedure is also presented to achieve the proportionality for iTAPAS. Numerical results obtained from applying the new and original algorithms for several large networks reveal that iTAPAS is nearly two times faster than TAPAS in achieving highly precise link flow solutions while it is practically identical to TAPAS in finding stable path flow solutions that meet consistency and proportionality.

1. Introduction

The traffic assignment problem (TAP) has been widely used to predict network flows that are an aggregate result of route choices of individual travelers under the user equilibrium (UE) principle (Wardrop, 1952), which states that for every origin-destination (O-D) pair all used paths have equal travel costs and no unused path has a lower cost. The UE-type traffic assignment problem was first formulated as a convex optimization problem with linear constraints by Beckmann et al. (1956). Substantial efforts have since been made to find efficient algorithms to solve this problem.

The first class of practical TAP algorithms operates in the space of link flows, known as link-based algorithms (Florian et al., 1987; Fukushima, 1984; LeBlanc et al., 1975). Because they are easy to implement and efficient in getting reason-

* Corresponding author at: A610 Ruth Mulan Chu Chao Bldg., 800 Dongchuan Rd., Shanghai 200240, China. Tel.: +86 (21) 3420-8385. Fax: +86 (21) 3420 6197.
E-mail address: chi.xie@sjtu.edu.cn, chi.xie@mail.utexas.edu (C. Xie).
[1] A823 Ruth Mulan Chu Chao Bldg., 800 Dongchuan Rd., Shanghai 200240, China. Fax: +86 (21) 3420 6197.

http://dx.doi.org/10.1016/j.trb.2016.08.009
0191-2615/

ably satisfactory solutions for supporting basic urban travel forecasting functions, link-based algorithms have been widely adopted in practice, virtually by all major software vendors. However, link-based solutions are not sufficiently detailed for many important applications, such as select link analysis (Bar-Gera et al., 2012; Boyce and Xie, 2013) and estimation of turning movements. Moreover, link-based algorithms are not well suited to deliver high levels of precision that are often critical to the utilization of more detailed solutions. Another class of algorithms, known as path-based algorithms (Florian et al., 2009; Jayakrishnan et al., 1994; Kumar and Peeta, 2011; Kumar et al., 2012; Larsson and Patriksson, 1992; Lee et al., 2003), address the above shortcomings. These algorithms not only generate and store detailed path solutions for each O-D pair, but they can also converge to optimum much more quickly than the link-based counterparts. Unfortunately, path-based algorithms scale poorly in general, thanks to the overhead involved in explicitly manipulating paths, which becomes a heavy burden in large-scale regional networks that involve millions of O-D pairs (Inoue and Maruyama, 2012).

In the past decade, solution techniques for TAP have been advanced substantially by the development of the bush-based algorithms (Bar-Gera, 2002; Dial, 2006; Gentile, 2012; Nie, 2010, 2012; Xie et al., 2013). These algorithms have attracted much attention because they promise highly converged and detailed TAP solutions for large-scale networks at modest computational costs. All bush-based algorithms known so far share two important features: (1) they construct and maintain a UE bush for each origin (or destination); (2) they assign demands originating at each origin only to the corresponding bush. The concept of bush was originally proposed by Dial (1971) in his seminal work on the logit-based traffic assignment problem for uncongested networks, albeit it had not been used for solving TAP until much later (Dial, 1999).

A primary difference between a tree and a bush is that the latter contains paired alternative segments (PAS). Formally, a PAS is defined as two completely disjoint path segments with the same starting and ending nodes. Thus, a PAS marks the distinct parts of two or more paths connecting one O-D pair. The idea of focusing only on these parts of a bush in TAP was formally introduced by Bar-Gera (2010), though its use for solving TAP has a much longer history. For example, the network simplex (NS) method adapted by Nguyen (1974) actually identifies and uses PASs, although it does not store them for repeated uses. Specifically, NS maintains a spanning tree for each origin such that a PAS can be detected by augmenting a non-tree link into the tree. Zheng (2015) recently revisited this algorithm and proposed a new variation of NS. Recently reported numerical results (Xie and Xie, 2015; Zheng, 2015) demonstrated the high efficiency of this algorithm. It is somewhat surprising that NS has not received more attention, given its great potential to achieve high performance (compared to its primary rival, the Frank-Wolfe (FW) algorithm, at the time when it was adopted for solving TAP in 1970s). Another example is Dial's (2006) Algorithm B, which employs PAS in a more direct way. Algorithm B performs the assignment between a special PAS that consists of the shortest and longest segments between two consecutive nodes. As similar to NS, these PASs in Algorithm B are constructed when needed, but not stored.

Bar-Gera (2010) systematically explored the use of PASs for solving TAP. His TAPAS algorithm has three notable innovations in utilizing PASs. First, it employs a breadth first search (BFS) method to identify PASs directly from the shortest path trees, which can be re-constructed easily in each iteration. In contrast, NS and Algorithm B need to maintain a spanning tree or a bush for each origin in the course of the algorithmic iterations, which usually requires extra memory space and computational efforts. Second, it stores PASs in a list in order to save the time spent on repeatedly identifying the same segments. Third, it connects multiple origins to one PAS by matching an origin to an existing PAS in the list, which helps secure a stable path solution by forcing proportionality (Bar-Gera et al., 2012; Boyce and Xie, 2013; Dial, 1999).

The literature has demonstrated the great potential of TAPAS in designing efficient TAP algorithms (Inoue and Maruyama, 2012; Xie and Xie, 2015). Yet, our experiments with the algorithm brought a few methodological and implementation issues to light, which motivated the work presented in this paper. The first issue lies in the BFS method for PAS identification. In this method, the higher cost segment of a PAS is chosen arbitrarily, which sometimes leads to suboptimal performance. Perhaps more critically, the BFS method may not always identify a PAS that ensures the improvement in the solution. In such cases, a branch shift procedure is needed to ensure the convergence, which complicates the algorithm implementation, as well as the convergence proof. The second issue is related to the many-to-one mapping between origins and PASs (i.e., one PAS is shared by many origins). Bar-Gera (2010) suggested that this strategy expedites the overall convergence since it bundles the assignment of various origins by enforcing the same proportionality on a shared PAS. It is unclear, however, whether or not the benefits from cross-origin proportionality adjustments can offset the resulting extra computational overhead.

In light of the above observations, an improved variant of TAPAS, dubbed iTAPAS, is developed in this paper. iTAPAS introduces a new PAS identification method that will eventually either identify a PAS with an equal cost on the two segments for the potential link or reduce the origin-specific flow of the potential link to zero by identifying more than one PASs for the same potential link as whenever needed. This property of the new PAS identification method is not only expected to fasten the speed of the UE convergence, but also leads to a new proof for the convergence of PAS-based algorithms (see Section 5). Moreover, iTAPAS associates each PAS with only one origin (i.e., the one-to-one mapping) to speed up the efficiency of the flow shift operations. Our numerical results show that both algorithmic improvements can significantly reduce the convergence time of TAPAS, and iTAPAS can be almost twice as faster as TAPAS in solving regional-scale networks. Since iTAPAS does not enforce proportionality across origins during the solution process, we propose to obtain such a solution by a post-process procedure. Our numerical result proves the effectiveness of the post-process in achieving consistency and proportionality while justifying that the difference of the convergence levels between iTAPAS and TAPAS is negligible. The last contribution is the proposition of several alternative types of optimality conditions based on segments or PASs (see Section 2), which well explain the merits of placing focus only on PASs in solving TAP.

The rest of this paper is organized as follows. Section 2 reviews the origin-based formulation for TAP, followed by several new types of optimality conditions. Section 3 reviews and discusses several key components of TAPAS. Section 4 presents the algorithmic procedure of iTAPAS. A new convergence proof of PAS-based algorithms is presented in Section 5. A post-process procedure for consistency and proportionality is summarized in Section 6. Section 7 reports and compares the convergence performance of the UE and proportionality properties for TAPAS and iTAPAS. For a more comprehensive comparison between iTAPAS and other eight origin-based algorithms in terms of their algorithmic and computational performance, interested readers are referred to Xie and Xie (2015). Finally, the last section concludes the paper with a summary on the major findings and our suggestions for future research.

2. Origin-based formulation and optimality conditions

Consider a directed transportation network $G(N, A)$ where N and A denote the sets of nodes and links, respectively. Suppose that G is strongly connected, by which we mean there is at least one path between any two nodes in G. N consists of a set of origins denoted by R and a set of destinations denoted by S. The travel demand departing from an origin $r \in R$ to a destination $s \in S$ is denoted by q_{rs}. Link $(i, j) \in A$ is directed from node i to node j.

It is well known that, under the above assumptions, the traffic assignment problem satisfying Wardrop's first principle can be formulated as follows (Bar-Gera, 2002; Nie, 2010):

$$\min z(x) = \sum\nolimits_{(i,j)\in A} \int_0^{x_{ij}} t_{ij}(w)\mathrm{d}w \tag{1}$$

$$\text{subject to} \quad \sum\nolimits_{l\in O(j)} x_{jl}^r - \sum\nolimits_{i\in I(j)} x_{ij}^r = q_j^r \qquad \forall j \in N,\ r \in R \tag{2}$$

$$\sum\nolimits_r x_{ij}^r = x_{ij}\ \ x_{ij}^r \geq 0\ \forall (i,j) \in A,\ \ r \in R \tag{3}$$

where x_{ij}^r is the flow on link (i, j) contributed by all O-D pairs r-s originating from origin r, and we have $\sum_{r\in R} x_{ij}^r = x_{ij}$; $O(j)$ and $I(j)$ denote the set of successor(s) and predecessor(s) of node j, respectively. Let x_{ij} be the flow traversing link (i, j) and $t(\cdot)$ be a continuous, differentiable, increasing and convex function of x_{ij}. q_j^r is then defined as follows:

$$q_j^r = \begin{cases} \sum\limits_{r\in S(r)} q_{rs} & \text{if } j = r \\ -q_{rs} & \text{if } j = s \\ 0 & otherwise \end{cases} \qquad \forall j \in N,\ r \in R,\ s \in S \tag{4}$$

The above formulation is defined in the origin-based link flow space with respect to each origin; this block-diagonal structure of constraints allows a natural decomposition of the problem with respect to origins. Specifically, we can separately consider the following problem for each origin r (Nie, 2010):

$$\min z_r(x) = \sum_{(i,j)\in A} \int_0^{x_{ij}^{r-}+x_{ij}^r} t_{ij}(w)\mathrm{d}w \tag{5}$$

$$\text{subject to} \quad \sum_{i\in I(j)} x_{ij}^r - \sum_{l\in O(j)} x_{jl}^r = q_j^r \qquad \forall j \in N \tag{6}$$

$$x_{ij}^r \geq 0\ \forall (i,j) \in A \tag{7}$$

where $x_{ij}^{r-} = \sum_{o\in R, o\neq r} x_{ij}^o$ are flow contributed by all O-D pairs o-s, where $o \neq r$. Let u_j^r be the multiplier associated with node j for origin r, then the Karush-Kuhn-Tucker (KKT) conditions of the problem in (5)–(7) includes the feasible conditions (6) and (7) and the following complementary conditions:

$$t_{ij}\left(x_{ij}\right) + u_i^r - u_j^r \geq 0 \quad \forall (i,j) \in A \tag{8}$$

$$x_{ij}^r\left[t_{ij}\left(x_{ij}\right) + u_i^r - u_j^r\right] = 0 \quad \forall (i,j) \in A \tag{9}$$

It is clear that u_j^r should be interpreted as the minimum travel cost from origin r to node j. The above conditions state that, at UE, a link (i, j) can receive positive flow departing from origin r if and only if (i, j) is part of a shortest path tree originated at r.

Definition 1. (Link reduced cost). The reduced cost π_{ij}^r for link (i, j) in the subnetwork rooted at origin r is defined as $\pi_{ij}^r = u_i^r + t_{ij}^r - u_j^r$.

Given the above definition of link reduced cost, the complementary conditions (8) and (9) can be converted into the following form, known as the optimality conditions for link reduced cost:

$$x_{ij}^r \pi_{ij}^r = 0 \quad \forall (i,j) \in A \tag{10}$$

$$\pi_{ij}^r \geq 0 \quad \forall (i,j) \in A \tag{11}$$

The above optimality conditions for link reduced cost state that, at equilibrium, any link with positive flow should have a zero reduced cost. This fact can be extended to a more general form by introducing the concept of segment and segment reduced cost in the following.

Definition 2. (Segment). Following the definition in Bar-Gera (2002), we define a simple path segment, denoted by e, as a sequence of distinct nodes and only one directed link is allowed between every pair of nodes. Hence a segment never contains cycles. We further assume that the following is always satisfied for any segment: all links in one segment have the same direction; the size of a segment, denoted by $|e|$, is the size of links it contains; link costs are additive along a segment and the cost of a segment, denoted by t_e, is the sum of all its link costs, i.e., $t_e = \sum_{(i,j)\in A_e} t_{ij}$. Given a path segment $e = [m, \ldots, n]$, we call its first node m as the tail, denoted by e_t, and call its last node n as the head, denoted by e_h. The set of segments over the network is defined as E.

Definition 3. (Segment reduced cost). The segment reduced cost π_e^r for e in the subnetwork rooted at origin r is defined as $\pi_e^r = t_e^r + u_{e_t}^r - u_{e_h}^r$.

We define the segment flow bound f_e^r as the total amount of flow from origin r that utilizes the segment e; that is

$$f_e^r = \min\{x_{ij}^r : \ (i,j) \in e\} \tag{12}$$

Proposition 1. (Optimality conditions based on segment reduced cost). *A feasible solution of the traffic assignment problem in (1)–(3) is optimal if and only if the following conditions are satisfied,*

$$f_e^r \pi_e^r = 0 \quad \forall e \in E, \ \forall r \in R \tag{13}$$

$$\pi_e^r \geq 0 \quad \forall e \in E, \ \forall r \in R \tag{14}$$

Proposition 1 is correct because the conditions in (10) and (11) are equivalent to the conditions in (13) and (14) given Definition 2 and Definition 3. Proposition 1 generalizes the optimality conditions based on link reduced cost to those based on segment reduced cost. The former can be viewed as a special case of the latter when the size of all segments is restricted to be one. It is worth noting the following facts regarding the two optimality conditions: First, based on the optimality conditions for segment reduced cost, one can reduce the objective function (1), by identifying a segment e that violates the conditions in (13) and (14), or that satisfies $f_e^r > 0$ and $\pi_e^r > 0$, and shifting flow from segment e to the shortest path segment from node e_t to node e_h, because the reduced cost of the shortest path segment is defined to be 0. However, the total number of segments is extremely large in the general case, and it is almost impossible to enumerate all of them to force the conditions in (13) and (14) satisfied. Second, for the optimality conditions for link reduced cost in (10) and (11), the total number of conditions to be maintained are bounded by $|A||R|$. Hence it is operationally feasible to examine them by scanning each link. But the challenge now is how to reduce the objective function value by adjusting flow among links while keeping them still feasible. As a result, it seems hard to find an effective approach to turn a feasible solution to be optimal by any one of the two above optimality conditions, unless we take advantage of both.

Corollary 1. *Given a link (i, j) that violates the conditions in (10) and (11),* i.e., $x_{ij}^r > 0$ *and* $\pi_{ij}^r > 0$, *called potential link, any segment that consists of link (i, j) and carries a positive amount of flow will violate the conditions in (13) and (14).*

Corollary 1 provides us with an operable way to collect all segments that violate the segment optimality conditions. For simplicity, we can only collect the branch which is defined as follows:

Definition 4. (Branch). The branch of link (i, j) is defined as the set of segments satisfying the following criterion: (1) it begins with the origin and ends with link (i, j); (2) it carries a positive amount of flow.

The definition of branch was given by Bar-Gera (2010). Basically, shifting flow from branch segments to the shortest path segment can decrease the reduced cost of the potential link, and thus decrease the objective function value, while maintaining the feasibility condition (6) and (7) of involved links. See Fig. 1 for an illustration of branch. In this toy network, node 1 is the origin and node 5 is the destination; the shortest path tree is highlighted by dashed line and the value on

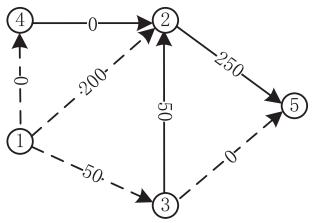

Fig. 1. Illustration of a potential link and its branch.

links represent the flow; we further assume that the network is not at equilibrium in that links (3, 2) and (2, 5) violate the optimality conditions (10) and (11). Basically, we call these two links potential links. For potential link (2, 5), its branch includes segment [1, 2, 5] and [1, 3, 2, 5], while segment [1, 4, 2, 5] is excluded from the branch because it carries zero flow.

Definition 5. (Paired alternative segments). Paired alternative segments (PAS) are defined as two completely disjoint segments with the same starting and ending nodes.

In a general network, instead of collecting branches, identifying PASs that incorporate the potential link and its (part of) branch as the higher cost segments would usually be much more efficient for two reasons: First, in most cases, identifying one or two branch segments would suffice to achieve the optimality of the potential link, it does not need to collect the whole branch; second, manipulating PASs apparently involves a less number of operations of links than manipulating branch segments.

Definition 6. (Equilibrated PAS). We call a PAS equilibrated, if (1) one segment of the PAS is the shortest path segment and (2) the other segment has an equal cost to that of the shortest path segment.

Proposition 2. (Optimality conditions based on PASs). *A feasible solution of the problem in (5)–(7) is an optimal solution if and only if for a given shortest path tree each used, non-tree link is part of at least one equilibrated PAS with respect to this shortest path tree.*

The above proposition establishes the equivalency relation between a UE solution and a certain set of PASs. Note that the number of PASs needed to maintain the equilibrium for a single-origin problem is bounded by the number of non-tree links, i.e., $|A| - |N|$. In practice, the total number of PASs maintained over the network is much less than $(|A| - |N|)|R|$ for two reasons: First, only some of non-tree links have a positive amount of flow at equilibrium. Second, PASs can be shared by links on different subnetworks. However, the total number of PASs that need to be identified in the solution equilibration process is typically much larger than $(|A| - |N|)|R|$, even as large as the number of O-D pairs. Most of these identified PASs are removed subsequently after use.

Given the PAS optimality condition, one remaining question is how to identify an effective PAS for each potential link (i, j). By effective in this paper, we mean that the origin-based flow and the reduced cost of link (i, j) can be decreased distinctly by shifting flow between segments of the PAS. Generally, we can define an effective PAS as follows.

Definition 7. (Effective PAS). A PAS $P(e_1, e_2)$ is an effective PAS for link (i, j) if (1) e_2 is part of a segment in the branch of link (i, j) and (2) e_1 is part of the current shortest path tree.

Usually, we regard a PAS as effective if its one segment is part of a branch segment and the other segment is part of the current shortest path. In practical applications, however, we prefer to identify a branch segment e_2 carrying more flow than another carrying less flow. The more flow the identified branch segment carries, the more flow can be decreased potentially for link (i, j). Rather, if the flow on segment e_2 is very low, say δ, then the decrease of flow on link (i, j) is limited to δ. As a result, the capability of a PAS identification method that can always find effective PASs plays a critical role in the convergence and efficiency of PAS-based algorithms.

3. Review of TAPAS

In this section, we first provide an implementable algorithmic procedure of TAPAS, which was only outlined in a concise conceptual form in its original article (Bar-Gera, 2010), and then place our focus on several primary operational elements of this procedure, including the PAS identification, PAS match and PAS flow operation. The related content to proportionality will be discussed in Section 6.

Procedure 1 TAPAS algorithm.

Input: Parameters ϵ, θ, ν, μ

Step 0: For each origin $r \in R$, initialize the subnetwork G_r as the shortest path tree rooted at r. Assign all flow to links on the tree. Update link travel time t_{ij} and its derivative t'_{ij} for each link $(i, j) \in A$. Create a set $P = \emptyset$ to keep PASs over the network, a set $P_j = \emptyset$ for each node $j \in N$ to keep PASs which head node is j, and a link set for each origin $L_r = \emptyset$, $r \in R$. A PAS is denoted by $p(e_1, e_2)$, and the beginning and ending nodes of it are denoted by p_t and p_h, respectively.

Step 1: For each origin r, do the following:

1.1 Remove all cycle flows on subnetwork G_r by the method described in Appendix B.

1.2 Update the shortest path tree T_r rooted at r; calculate the reduced cost π^r_{ij} for each non-tree link $(i, j) \in G_r \backslash T_r$, and if $x^r_{ij} > \epsilon$ and $\pi^r_{ij} > \theta$, push (i, j) into L_r.

1.3 While $L_r \neq \emptyset$, take a link (i, j) out of L_r, and repeat the following steps:

1.3.1 PAS match: Search every PAS $p(e_1, e_2) \in P_j$, push the origin r into the origin list of $p(e_1, e_2)$ and repeat Step 1.3 if $p(e_1, e_2)$ satisfies the following conditions; otherwise, go to Step 1.3.2.

(1) $t_{e_2} - t_{e_1} > \mu\pi^r_{ij}$;

(2) $f^r_{e_2} > \nu x^r_{ij}$;

(3) $(i, j) \in e_2$.

1.3.2 PAS identification: Identify a new PAS for link (i, j) with the BFS method. If a new PAS fails to be identified for (i, j), go to Step 1.3.3; else if a new PAS is identified successfully, shift flow from e_2 to e_1 once to equalize the cost of them: eliminate $p(e_1, e_2)$ immediately if $f_{e_2} = 0$ after the flow shift; otherwise, push $p(e_1, e_2)$ into P and P_j. Repeat Step 1.3.

1.3.3 Branch flow shift: Search the branch set of link (i, j), shift flow from these segments to the shortest path segment between r and j, until $x^r_{ij} = 0$ or $\pi^r_{ij} = 0$ is satisfied. Repeat Step 1.3.

1.4 Local PAS flow shift: Randomly choose a small number of PASs (i.e., 100 for small networks and 400 for large networks) from P and do flow shift operations for them to fasten the convergence of the algorithm.

Step 2: Repeat the following steps for 20 times.

For each PAS $p(e_1, e_2) \in P$, do the following

2.1 If the total segment flow bound $f_{e_1} = 0$ or $f_{e_2} = 0$, and $t_{e_1} \neq t_{e_2}$, eliminate $p(e_1, e_2)$ from P and P_j.

2.2 Shift flow from the higher cost segment to the lower cost segment to equalize their cost.

2.3 Redistribute flow among origins by the proportionality condition.

Step 3: If the convergence criterion is achieved, go to Step 4; otherwise, go to Step 1.

Step 4: Do a post-process for proportionality by Procedure 6.

Procedure 1 describes the complete algorithmic steps of TAPAS. ϵ and θ are the minimal flow and cost criteria, respectively. ν is the so-called flow effective factor used to restrict that the selected PAS segment e_2 should at least carry flow in the amount equal to νx^r_{ij}, or $f^r_{e_2} \geq \nu x^r_{ij}$. Hence $\nu = 0.25$ is recommended. μ is the cost effective factor which is used only in Step 1.3.1, to make sure that the segment cost difference of the matched PAS for (i, j) should at least be $\mu\pi^r_{ij}$, or $t_{e_2} - t_{e_1} \geq \mu\pi^r_{ij}$. $\mu = 0.5$ is recommended. Recall the definition of an effective PAS (see Definition 5), the relationship $t_{e_2} - t_{e_1} \geq \pi^r_{ij}$ is always valid for a newly constructed PAS if its segment e_1 is defined as part of a shortest path segment. In the practical implementation of TAPAS, it is usually not necessary to require the least cost segment is always exactly the shortest path segment. Instead, a segments cost difference restraint $t_{e_2} - t_{e_1} \geq \mu\pi^r_{ij}$ can be imposed to keep the relative effectiveness of flow shift operations. Moreover, by relaxing the effectiveness of a PAS with the cost effective factor μ, TAPAS needs to match an existing effective PAS for link (i, j) before constructing a new one for it, termed the PAS match in Step 1.3.1, which saves considerable computational time in identifying new PASs. In Step 0, P_j is created for convenience of searching existing PASs for link (i, j) in the PAS match step. In Step 1.1, removing cycle flows from the subnetworks is essential for the convergence and efficiency of TAPAS, for which an alternative method is provided in Appendix B. Next, we will carefully examine several key blocks of the above procedure followed by our remarks.

NS and Algorithm B identify PASs based on spanning trees and bushes, respectively. With the BFS method, TAPAS constructs new PASs directly based on the shortest path trees. The superiorities of the latter way are easy to enumerate, including without need of maintenance of bushes or spanning trees and without concern of possible standstill in algorithm convergence caused by the degenerated update of trees or bushes.[1] To construct a new PAS for link (i, j) that satisfies condition $x^r_{ij} > \epsilon$ and $\pi^r_{ij} > \theta$ the least cost segment e_1 is defined as the shortest path segment from node p_t to p_h, where $p_h = j$, and p_t is to be determined in searching the higher cost segment e_2. One way to perform the search is shown in Procedure 2.

In most cases, the above procedure can identify an effective PAS that satisfies $e_2 - e_1 \geq \pi^r_{ij}$ and $f_{e_2} \geq \nu x^r_{ij}$ in an acyclic subnetwork. If we define a topology order from node j to all the other nodes in the subnetwork, then the BFS method actually tries to scan as many links as possible in the current order level before going deeper. Such a property of BFS seems to help find the higher cost segments with fewer links, which is obviously preferable. One critical issue of the BFS method is its possible failure of identifying an effective PAS in the following two situations: (1) Cycle flows may appear in flow shifts for a new PAS in Step 1.3.2 of Procedure 1, and when this happens, it may lead to a failure of identifying a PAS for the rest links in L_r; (2) a flow effective factor ν is imposed to ensure that the flow of each link going in segment e_2 is

[1] It is well known that such a degeneration phenomenon is often seen in the network simplex method. We also notice that a similar degeneration effect occurs in bush-based algorithms when the maintained bushes are not well redefined.

Procedure 2 PAS identification method in TAPAS: BFS procedure.

Input: A shortest path tree rooted at origin r, where q_k stores the preceding node of node k in the shortest path tree, and Q_k stores the foregoing node of node k in the higher cost segment s_2. Set a node scan status variable $l_k = 0$ for each $k \in N$. Set a queue $D = \emptyset$ and identify a link (i, j) ready for identifying a new PAS.
Step 1: Visit each node k on the shortest path from node j backward to the origin r following q_k, and set their l_k as -1. Push back the node i into D.
Step 2: While $D \neq \emptyset$, do the following:
2.1 Take a node n from the front of D. If $l_n = -1$, then a new PAS is identified. Define $p_t = n$ and $p_h = j$, and go to Step 2.2. Then, if $l_n = 0$, set $l_n = 1$, and search each link (m, n) in the backward set of node n that $m \in I(n)$: set $Q_m = n$ and push m into the back of D if $x^r_{mn} > \nu x^r_{ij}$ and m is not in D, go to Step 2; if $l_n = 1$, go to Step 2.
2.2 Define the less cost segment e_1 from p_h to p_t following q_k; define the higher cost segment e_2 from p_t to p_h following Q_k. Push the origin r into the origin list of the new identified PAS. Stop the search procedure.

at least as large as νx^r_{ij} in BFS, where in case the flow spreads over many paths and no path segment satisfies $f^r_e \geq \nu x^r_{ij}$, the PAS will be not identified as well. To achieve the expected convergence, TAPAS tries to identify the branch segments of link (i, j) instead when the above two situations occur and to shift flow from these branch segments to the shortest path segment. The branch shift is actually a compromising approach for keeping validity of BFS, and it further complicates the implementation of TAPAS. In the next section, we propose a new PAS search method that not only can settle all above concerns, but also can lead to more "local" PASs. With this new search method, we even do not need the cycle-removing step, Step 1.1 of Procedure 1.

Prior to identifying a new PAS for link (i, j) using the BFS method, all existing PASs ending at node j are checked if one of them is effective for (i, j) in Step 1.3.1 of Procedure 1. It is noted that when we say that a PAS is effective for link (i, j), we exactly mean that this PAS is effective only for link (i, j) at the current origin r. At the PAS match step, when an existing PAS, which was originally created at one other origin, say r^0, is matched at the current origin r, this PAS thus is effective for these two origins. TAPAS stores all the matched origins into an origin list associated with each PAS for two purposes: (1) One is to realize proportionality, which we will discuss in Section 6; (2) the other one is to expedite the overall convergence of the algorithm, since it bundles the assignment of all origins associated with this PAS together and allows more flow to shift from segment e_2 to segment e_1. For the latter purpose, however, our research finding seems to reveal an opposite result; that is, performing cross-origin flow adjustments would actually slow down the convergence. To make it clear, we will show below how TAPAS adjusts the segment flow for a PAS.

Procedure 3 Flow shift operation in TAPAS.

Input: A PAS $p(e_1, e_2)$ where s_2 is the higher cost segment, and the origin list associated with this PAS is denoted by R_p.
Step 1: Update the segment costs t_{e_1} and t_{e_2}.
Step 2: For each origin $r \in R_p$, determine the segment flow bound $f^r_{e_2} = \min\{x^r_{ij},\ (i, j) \in s_2\}$, and the total segment flow bound $f_{e_2} = \sum_{r \in R_p} f^r_{e_2}$.
Step 3: Compute the total flow shift $\delta = \min[(t_{e_2} - t_{e_1})/(\sum_{(i,j)\in e_1} t'_{ij} + \sum_{(i,j)\in e_2} t'_{ij}),\ f_{e_2}]$. The amount of shifted flow for each origin is proportional to the total shifted flow δ, i.e., $\delta^r = (f^r_{e_2}/f_{e_2})\delta$.
Step 4: For each origin $r \in R_p$: Update $x^r_{ij} = x^r_{ij} + \delta^r$ and $x_{ij} = x_{ij} + \delta$ for each link $(i, j) \in e_1$; update $x^r_{mn} = x^r_{mn} - \delta^r$ and $x_{mn} = x_{mn} - \delta$ for each link $(m, n) \in e_2$.
Step 5: Update the link travel time t_{ij} and its derivative t'_{ij} involved in both segments.

Note that the maximum flow allowed for adjustment is actually the sum of flow generated from all origins in the associated list of the PAS. Hence, the flow shift operation for a PAS in TAPAS involves segment links in all subnetworks associated with this PAS. It is worth investigating whether or not this effort is paid off from the perspective of convergence efficiency[2] for several reasons. First, associating one PAS to multiple origins is not necessary for convergence. If we can keep that the higher cost segment carries an amount of flow larger than 0, indicating that the equalization purpose of two segments can be always satisfied, then how many and which origins are associated with this PAS would not make difference. Second, as seen in Procedure 3, the PAS flow shift operation by TAPAS requires visiting all corresponding segment links for each origin in the relevant list. As an illustration, the computation results for the Chicago Regional network show that the average number of origins associated with one PAS is about 41 and the maximum one is 1182. The PAS flow shift operation in this case becomes unnecessarily expensive, especially considering that this is the most frequently repeated operation in TAPAS.

Technically speaking, TAPAS is the most sophisticated and efficient algorithm design for TAP in the existing literature. This is largely because of its innovative use of PAS (for convergence and proportionality). In the following, we will show more advanced use of PAS for TAP in both technical and theoretical aspects, which leads to a more efficient and compact version of TAPAS, iTAPAS.

[2] We note that the multi-origin association relationship is also closely related to proportionality, which is discussed in detail in Section 6.

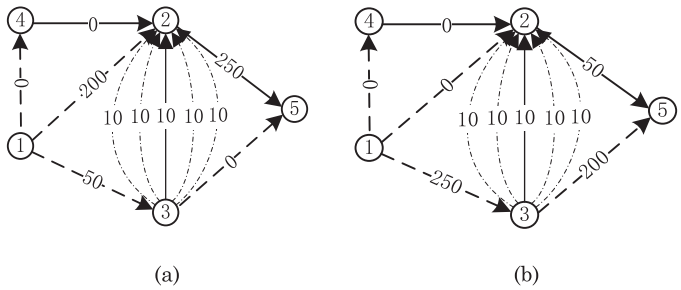

Fig. 2. Example network for an illustration of flow-spreading effects.

4. iTAPAS

We are now ready to present the iTAPAS algorithm. The first key improvement of iTAPAS against TAPAS is the proposition of a new PAS identification method, named maximum-flow first search (MFS). Just as its name implies, MFS identifies the higher cost segment always by using the maximum-flow link in the backward set of the node that starts from i back to the origin r, until the first node that is used by the corresponding shortest path as well. The steps of the MFS procedure are summarized as follows.

Except the key feature of the MFS method that identifies the higher cost segment by preferentially using the maximum-flow link in the backward set, we here want to emphasize the feedback mechanism realized in Step 2.2, which ensures that the flow on the potential link can be decreased effectively by identifying more than one PASs whenever it needs. The feedback mechanism is explained with the help of the example network shown in Fig. 2, where the situation is somehow extreme but it does happen in general networks. The toy network shown in Fig. 2(a) is defined as same as Fig. 1 except that there are four more segments (represented by dot dashed lines) connecting node 3 and node 2, and the flow is equally split over all five segments (links). In such case, the MFS method would identify [1, 2, 5] as the higher cost segment firstly. If we further assume that the potential link, (2, 5), still violates the optimality condition (10) and (11) after we shift all 200 units of flow to the shortest path [1, 3, 5], shown in Fig. 2(b), then the MFS method will continue to identify segment [1, 3, 2, 5] with solid lines or with dot dashed segments until the optimality conditions are satisfied for the potential link in the current flow pattern.

As noted by Bar-Gera (2010) in Section 6.3 of his paper, the flow-spreading case shown in Fig. 2(a) would cause improper flow assignment problems, possibly leading to the non-convergence when the BFS method is used. Let us explain this phenomenon by using the same example. Similarly, in Fig. 2(a), the BFS method will identify [1, 2, 5] as the higher cost segment firstly if a flow effective factor $\nu = 0.25$ is imposed. Then, in Fig. 2(b) the BFS method fails to identify any PAS for potential link (2, 5) due to the restriction of $\nu = 0.25$. As a remedy to the BFS method and as a necessary supplement to the convergence proof, Bar-Gear (2010) proposed for TAPAS to collect the branch set for potential links that fail to identify a PAS successfully, and to shift flow from branch segments to the shortest path. The branch shift operation complicates TAPAS to some extent.

Moreover, the acyclicity of origin-based subnetworks is not required by the MFS method, because it can identify and remove cycle flow that hamper the search process of effective PASs. This feature is illustrated by the example in Fig. 3, where the two subnetworks are different only in the amount of flow on the directed cycle [3, 5, 6]. Links that are not on the shortest path tree are marked by solid lines, while the dashed are the shortest path links. Numbers on links are origin-based link flow values. Now let us identify a PAS for link (3, 4) in both cases by using Procedure 4, which is supposed to

Procedure 4 MFS procedure.

Input: A shortest path tree rooted at origin r, where q_k stores the preceding node of node k in the shortest path tree, and Q_k stores the foregoing node of node k in the higher cost segment e_2. Set a node scan status variable $l_k = 0$ for each $k \in N$. A potential link (i, j) ready for identifying a new PAS.

Step 1: Visit each node k on the shortest path from the node j back to the origin r following q_k, and set their l_k as -1. Define the next scanning node $n = i$.

Step 2: Search each backward link (m, n) of node n, let $\hat{m} = \arg\max\{x^r_{mn} : m \in I(n)\}$, store $Q_{\hat{m}} = n$. There are three cases, according to the status of $l_{\hat{m}}$: If $l_{\hat{m}} = -1$, define $p_t = \hat{m}$, $p_h = j$ and go to Step 2.1; else if $l_{\hat{m}} = 1$, then a directed cycle is identified and go to Step 2.3; else if $l_{\hat{m}} = 0$, set $n = \hat{m}$ and $l_{\hat{m}} = 1$ and repeat Step 2.

2.1 Define the lower cost segment e_1 from p_h back to p_t following q_k, and define the higher cost segment e_2 from p_t to p_h following Q_k. Go to Step 2.2.

2.2 Shift flow from the higher cost segment e_2 to the lower cost segment e_1. If the potential link (i, j) satisfies the conditions in (10)-(11) after the flow shift operation, a new PAS is identified successfully, stop the procedure; otherwise, go to Step 1 to invoke the MFS method one more time to identify a new PAS for the potential link (i, j).

2.3 Find out the cycle w following $Q_{\hat{m}}$. Set $\delta = \min\{x^r_{mn} : (m, n) \in w\}$, then update $x_{mn} = x_{mn} - \delta$, $x^r_{mn} = x^r_{mn} - \delta$; update t_{mn} and t'_{mn} of links on the cycle. Go to Step 1.

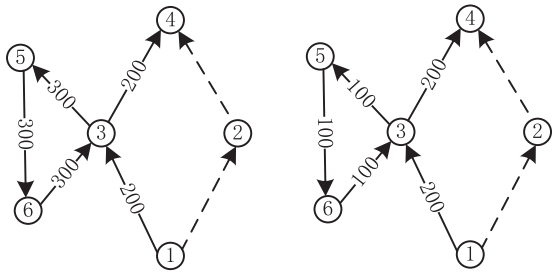

Fig. 3. Example network for the cycle-related issues in searching a PAS.

choose the higher cost segment always with the "most used link" in the backward set. Considering Fig. 3(a), we then know that the node search sequence would be 4, 3, 6, 5 and 3, and a directed cycle $w = [3, 6, 5]$ with a cycle flow of 300 is identified; by removing the cycle and re-searching from the beginning, we obtained a higher cost segment $e_2 = [4, 3, 1]$ with a flow of 200. Consider the Fig. 3(b), the MFS method examines nodes in the order of 4, 3 and 1, and the same higher cost segment $e_2 = [4, 3, 1]$ with a flow of 200 is identified. The MFS method leaves the cycle unchanged in Fig. 3(b), where the cycle flow is smaller than the flow of link (1, 3) and it thus would not impact the PAS identification result.

The MFS method is also expected to find a short higher cost segment, provided with the basic cognition that more flow tends to be assigned to shorter paths. This expectation is further verified in Section 7 with numerical results. On the other hand, the MFS method tends to identify higher cost segments with more segment flows than that on segments identified by the BFS method (see the numerical statistics on segment flows in Section 7).

Parameters ϵ, θ, ν, and μ in iTAPAS are defined exactly the same as those in TAPAS, while we should note that the flow and cost effective factors are used only in the PAS match step here. In Step 1.2.1, when an existing PAS p_m is matched for a potential link (i, j), rather than associating the current origin r to p_m, we choose to equalize the segment cost for p_m by shifting the flow from its original origin r_0. If the origin-specific segment flow bound on one segment, i.e., $f_{e_2}^{r_0}$, reduces to 0 before link (i, j) satisfies the reduced cost optimality conditions, it will further identify a new PAS for (i, j) in the PAS identification step. This design not only leads to the fact that only one unique origin is associated with each PAS, but also may be able to cut down the flow-spreading effect in the convergence stage for the efficiency purpose. It is worth noting that the link costs are updated whenever the flows on these links are changed. One tricky thing here is that whenever we update the link cost, the shortest paths may be accordingly changed and updating the shortest paths subsequently becomes unacceptably costly. In our implementation, whenever the link flows are changed, the link costs are updated but the shortest paths are not until the next iteration. This compromise may lead to missing identification of some effective PASs in the current iteration, which, though, is not a big problem because these missing PASs will be identified in the subsequent iterations. In Step 2.1, if the flow on an origin-based segment of a PAS reduces to zero while it is not equilibrated yet, we will match several adjacent origins for this PAS trying to equalize this PAS before we eliminate it from the list. Note that this strategy guarantees that every eliminated PAS will be scarcely reused in the subsequent iterations. The operations for the proportionality in Step 4 are discussed in Section 6.

The flow shift operation in iTAPAS is more straightforward than that in TAPAS, as there is only one origin r associated with each PAS. Specifically, we have $f_{e_2} = f_{e_2}^r$; given $\delta = \min[(t_{e_2} - t_{e_1})/(\sum_{ij \in e_1} t'_{ij} + \sum_{ij \in e_2} t'_{ij}),\ f_{e_2}]$, shifting δ directly for origin r accomplishes the PAS flow shift operation. This flow shift mechanism is based on the Newton method and is inherited from Dial (2006).

A relatively simple implementation of TAPAS or iTAPAS can work very well for solving small- and medium-sized networks, as the number of PASs in such cases is relatively small and quite easy to handle. However, as the number of PASs increases up to tens of thousands when solving large regional networks, managing such a list of PASs is not trivial and is critical to the success of the algorithm. According to our limited experience, several algorithmic settings need to be treated rigorously for solving large-scale networks. First, whenever a new PAS is identified, do the flow shift operation first before adding it into the list. This "shift first" rule will save considerable efforts in managing the PAS list by preventing many "shortly lived" PASs from being added into the list, especially in the early stage of the equilibration process. Second, the flow shift operation within a PAS is the most frequently repeated steps in TAPAS and iTAPAS. Shifting flow for each PAS in the list without considering how well it has converged is not only costly, but also might be needless. If the convergence level over the whole network is not high, for example, the relative gap (RG) is 10^{-4}, then a PAS with a segment cost difference of 10^{-10} is usually not worth a flow shift operation, of which the contribution in the current iteration can be negligible. Therefore, it is recommended to check whether or not a parameter λ is needed to shift flow for a PAS. In the current implementation, we set $\lambda = \text{RG} / 1000$ for both TAPAS and iTAPAS. Third, the flow precision ϵ and cost precision θ need to be set properly. If they are set as relatively large values, the algorithm may fail to converge to an expected precision, because the algorithm would not identify PASs for links with a flow smaller than ϵ, or with a reduced cost smaller than θ; if they are

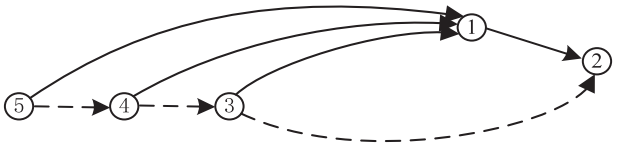

Fig. 4. Example network for an illustration of integration of several PASs.

set as very small values, the effort of searching PASs for such links would be unnecessary. In our implementation, ϵ and θ are set as 10^{-12} and 10^{-16}, respectively.

5. Proof of convergence

Several analytical results are presented here to justify the convergence performance of the proposed iTAPAS. First, as shown in the previous sections, since iTAPAS discards the branch operations, we first need to prove that iTAPAS can converge to the UE solution without the branch operations. Second, the new proof shows that the MFS method will eventually either identify a PAS with an equal cost on its two segments for each potential link, or reduce the origin-specific flow of the potential link to zero, which to our best knowledge is the most effective way to decrease the objective function value for each potential link.

Lemma 1. *Given a potential link (i, j) at origin r, the MFS method guarantees that each identified PAS $p(e_1, e_2)$ for the potential link at least satisfies $f_{e_2} \geq \tau \cdot \eta_{min}$ and $t_{e_2} - t_{e_1} \geq \pi^r_{ij}$ where $\eta_{min} = \min_{j \in N \setminus \{r\}} \{\sum_{i \in I(j)} x^r_{ij} > 0\}$ and τ is a positive constant.*

Proof. According to Procedure 4, all nodes scanned by the MFS method must have a flow higher than zero because of the flow conservation constraint. Any link (i, j) that consists a higher cost segment e_2 must share the maximum flow among all links into node j, i.e., $x^r_{ij} = \max_{m \in I(j)} x^r_{mj}$. If we define the minimum link flow over the network except for the origin as $\eta_{\min} = \min_{j \in N \setminus \{r\}} \{\sum_{i \in I(j)} x^r_{ij} > 0\}$ (where it is noted that the minimum is evaluated over all nodes in N except for origin r), then any link in a higher cost segment must carries an amount of flow higher than $\tau \cdot \eta_{\min}$, where τ is the reciprocal of the largest number of links entering into nodes. Thus we have $f_{e_2} \geq \tau \cdot \eta_{\min}$. On the other hand, according to Definition 3, we have $\pi^r_{e_2} = u^r_{p_t} + t_{e_2} - u^r_{p_h}$ and $\pi^r_{e_1} = u^r_{p_t} + t_{e_1} - u^r_{p_h}$, and we further have $\pi^r_{e_2} - \pi^r_{e_1} = t_{e_2} - t_{e_1}$ by making a subtraction. Because the lower cost segment e_1 is defined as the shortest path segment when it is created, $t_{e_2} - t_{e_1} = \pi^r_{e_2} \geq \pi^r_{ij}$ exists. The proof is finished. □

Lemma 2. *The MFS method will eventually either identify a PAS with an equal cost on the two segments for the potential link (i, j), or reduce the origin-specific flow x^r_{ij} of the potential link to zero.*

Proof. Following Lemma 1, Lemma 2 is forced to be valid by the feedback mechanism (Step 2.2 of the Procedure 4), which allows the MFS method to identify more than one PAS for the potential link. As noted by Lemma 1, the origin-specific flow x^r_{ij} can be always reduced to zero by identifying a finite number of PASs with MFS. As a result, without loss of generality, let us suppose that $n+1$ PASs, $n \geq 0$, were identified orderly for the potential link by the MFS method. Note that if $n+1$ PASs are identified by the MFS method, then the flow on the higher cost segment of any of the first n PASs should reduce to zero, and three situations arise for the $(n+1)$th PAS: (1) The flow on the higher cost segment is larger than zero, and the two segments are equalized; (2) the two segments are not equalized, but the origin-specific flow on the potential link reduces to zero; (3) the two segments are equalized, while the origin-specific flow on the potential link reduces to zero. The proof is finished. □

Remark 1. Lemma 2 in fact can be equivalently stated as follows: Following Lemma 1, the PASs identified and operated by the MFS method for the potential link constitute a subnetwork conforming to the UE conditions locally. This result can be intuitively illustrated by the toy network shown in Fig. 4, which is designed as an integration of three PASs associated to the potential link (1, 2), these three PASs are $P1$(3, 2; 3, 1, 2); $P2$(4, 3, 2; 4, 1, 2); $P3$(5, 4, 3, 2; 5, 1, 2). The dashed link represents part of the relevant shortest path tree. Without loss of generality, let us suppose that the MFS method found and handled these three PASs orderly for the potential link (1, 2). As argued above, the first two PASs, $P1$ and $P2$, must carry no flow on their higher cost segments, because the third PAS, $P3$, is identified. The two segment costs of $P3$ will be equalized, or they are not equalized, but the origin-specific flow on link (1, 2) reduces to zero. In any of the above two cases, the flow pattern on the subnetwork satisfies the UE conditions if only this subnetwork is considered.

Lemma 3. *Following Lemmas 1 and 2, the reduction in the objective function by implementing an MFS operation on a potential link (i, j) is positive and given by the following formulation:*

$$\Delta = \sum_{(i,j) \in A_s} \int_0^{x^0_{ij}} t_{ij}(w)\mathrm{d}w - \sum_{(i,j) \in A_s} \int_0^{x^*_{ij}} t_{ij}(w)\mathrm{d}w > 0 \tag{15}$$

where A_s is the set of links pertaining to the PASs identified by the MFS method, x_{ij}^0 is the link flow before the MFS method is implemented, and x_{ij}^* is the link flow after the MFS method is implemented.

Proof. The above lemma is clearly correct given the following two facts: (1) Shifting flow from the higher cost segments to the least cost segments changes the objective function value contributed by the subnetwork constituted by the links in A_s, and we have $\sum_{(i,j)\in A_s}\int_0^{x_{ij}^*} t_{ij}(w)dw = \min\sum_{(i,j)\in A_s}\int_0^{x_{ij}} t_{ij}(w)\mathrm{d}w$, where $x_{ij} \geq 0$; (2) the objective function value contributed by the links in set A/A_s is unchanged. □

Theorem 1. *If $\Phi_k \to 0$ and in every iteration k the MFS method is applied for every link with $x_{ij}^r > 0$ and $\pi_{ij}^r > \Phi_k$, then the sequence of objective function value $z(\boldsymbol{x}^k)$ converges to UE.*

Proof. Given the results in Lemmas 1, 2 and 3, the above theorem can be proved by using the same method of Theorem 1 in Bar-Gera (2010). □

6. Post-process for consistency and proportionality

This section presents a post-process aimed at forcing a UE solution to satisfy the condition of consistency and proportionality. The post-process can be viewed as a simplification of the procedure proposed in Bar-Gera (2010), which tries to maximize the entropy (locally) based on the set of PASs collected in each iteration. Following Bar-Gera (2010), we first define the conditions of consistency and proportionality as follows.

Definition 8. (Consistency). Given a well converged UE solution that the reduced cost of any used segment is less than a small value, say $\varepsilon = 10^{-12}$, then for any given O-D pair *r-s*, the set of paths with a reduced cost less than ε is uniquely determined and called the εUE path set. Further, we say that the εUE path set is under consistency condition if the demand of O-D pair *r-s* are distributed over all paths in the path set without violating the UE conditions.

Definition 9. (Proportionality). At the εUE solution of the problem in (1), which is further assumed to satisfy the consistency condition, for any given PAS $p(e_1, e_2)$, the proportionality condition requires that the proportion of flow on the two segments e_1 and e_2 should be the same for all relevant origins or destinations.

To our current knowledge, a path flow solution yielding entropy maximization definitely satisfies consistency and proportionality, while the reverse is not necessarily correct (Bar-Gera and Boyce, 1999). However, existing evidence shows that the difference between them in practice is relatively small and their resulting path flow solutions can be viewed as being practically equivalent (Bar-Gera, 2006). If this result is considered, achieving consistency and proportionality based on PAS seems to be preferable than seeking entropy maximization, because most existing methods (Bar-Gera, 2006; Bell and Iida, 1997; Kumar and Peeta, 2015; Rossi et al., 1989) for deriving entropy maximization require the UE path set as input, which is typically not easy to obtain.

Although the consistency condition given in Definition 8 is defined in terms of path flows, it can be equivalently defined in the origin-based link flow space. Specifically, the consistency condition holds if the following conditions are satisfied for each link in each subnetwork G_r^*, $r \in R$, without violating the UE conditions.[3]

$$\begin{cases} x_{ij}^r > \omega & \text{if } \pi_{ij}^r < \varepsilon \\ x_{ij}^r < \omega & \text{if } \pi_{ij}^r > \varepsilon \end{cases} \tag{16}$$

where G_r^* is the origin-based subnetwork that aggregates all links on the UE paths connecting all destinations and a specific origin *r*; ω is the flow accuracy recognized by the algorithm. The above conditions state that each link in G_r^*, $r \in R$, should carry positive flow, unless the use of the link causes the UE violation. When a subnetwork is not in consistency, one can, of course, seek to improve the efficiency by collecting links that violate the conditions in (16), which are:

$$x_{ij}^r > \omega \text{ and } \pi_{ij}^r > \varepsilon \tag{17}$$

$$x_{ij}^r < \omega \text{ and } \pi_{ij}^r < \varepsilon \tag{18}$$

Since links satisfying the condition in (17) violate the UE conditions, a well converged UE solution should not contain such links.[4] For a link (i, j) satisfying the conditions in (18), one can increase its flow from zero to a positive value without violating the UE conditions by operating a relevant PAS. To this end, one should identify a PAS $p(e_1, e_2)$ satisfying the following two conditions: (1) Its segment e_1 incorporates (i, j); (2) at least two origins r and r' are associated with this PAS to satisfy $f_{e_2}^r > \omega$ and $f_{e_1}^{r'} > \omega$. Then one can shift at origin r a certain amount of flow $0 < \delta < \min(f_{e_2}^r, f_{e_1}^{r'})$ from e_2 to e_1,

[3] In some extreme cases, $x_{ij}^r = 0$ and $\pi_{ij}^r < \varepsilon$ hold in the consistency condition, because an increase of flow on link (i, j) will incur the UE violation.

[4] The set of PASs collected in the UE-finding process can ensure that the condition in (17) is always satisfied, but it is not sufficient for the condition in (18).

and shift at origin r' the same amount δ from e_1 to e_2 correspondingly to increase the flow on x_{ij}^r from zero to δ. If we can always find such a PAS for each link that satisfies the conditions in (18), then the consistency condition will be assured, and the set of these PASs is consistent.[5] Unfortunately, no criterion that guarantees us to find a consistent set of PASs exists in the current literature and any effort on solving this problem definitely deserves a new paper.

If a consistent set of PASs is given, seeking proportionality is relatively easy. For any PAS $p(e_1, e_2)$ in the set, let us assume that every origin r is associated with PAS p if its corresponding εUE subnetwork G_r^* contains this PAS and the associated origin list is denoted by R_p. For each origin $r \in R_p$, we compute the segment flow as follows,

$$g_e^r = \eta_{e_h}^r \prod_{(i,j)\subseteq e} \phi_{ij}^r \tag{19}$$

where $\eta_{e_h}^r$ denotes the total flow arriving at the ending node of segment e in G_r^*, where $e = e_1$ or e_2; ϕ_{ij}^r is the link proportion, i.e., $\phi_{ij}^r = x_{ij}^r / \sum_{(i,j)\in I(j)} x_{ij}^r$. It is noted that the path segment flow calculated by (19) satisfies, by default, the proportionality condition among destinations for the same origin r. In other words, the proportion of flows between the two segments of a PAS is always the same regardless their different destinations in G_r^*. However, the proportions among origins are not necessarily the same and additional computational efforts are further required to equalize them.

The proportions among origins can be equalized by redistributing segment flows for each PAS in the consistent set. For a given PAS $p(e_1, e_2) \in P*$, we can calculate the origin-specific proportion and the PAS proportion by the following formulations, separately,

$$\rho^r = \frac{g_{e_1}^r}{g_{e_1}^r + g_{e_2}^r} \quad \forall r \in R_p \tag{20}$$

$$\rho = \frac{\sum_{r\in R_p} g_{e_1}^r}{\sum_{r\in R_p} \left(g_{e_1}^r + g_{e_2}^r\right)} \tag{21}$$

where segment flow $g_{e_1}^r$ and $g_{e_2}^r$ are computed by the formula in (19). Our goal is to equalize the values of ρ^r and ρ by adjusting a certain amount of flow δ^r for each origin, while making sure that $\sum_{r\in R_p} \delta^r = 0$ holds all the time. To this end, the origin-specific flow deviation δ^r is computed as follows,

$$\delta^r = \rho\left(g_{e_1}^r + g_{e_2}^r\right) - g_{e_1}^r \quad \forall r \in R \tag{22}$$

We then need further to set $x_{ij}^r = x_{ij}^r + \delta^r$, $\forall (i,j) \in e_1$, and $x_{ij}^r = x_{ij}^r - \delta^r$, $\forall (i,j) \in e_2$, for each relevant origin.

The flow adjustment among origins is rather straightforward and can be accomplished by only one iteration of the above naive method for an "isolated PAS", but it would be more challenging when the PAS is not "isolated" . One can judge whether one PAS is isolated or not by checking its segment flows: A PAS is isolated if $f_{e_1}^r = g_{e_1}^r$ and $f_{e_2}^r = g_{e_2}^r$ for each of its associated origins; otherwise, this PAS is not isolated. Bar-Gera (2010) provided a formal evaluation to show that the proportionality condition can be achieved by the above "naive approach" even the PAS is not isolated, although multiple iterations may be required (see Section 7 in his paper). He further suggested to use the quadratic approximation to capture the nonlinear variations of g_e^r. However, our experience shows that the quadratic approximation approach fails to keep the condition of $\sum_{r\in R_p} \delta^r = 0$ all the time, so the naive approach is chosen. The detailed procedure of the post-process for consistency and proportionality can be found in Appendix B.

7. Numerical experiments

In this section, we compare the convergence performance of TAPAS and iTAPAS through a set of numerical experiments. To provide a performance benchmark, the Frank-Wolfe algorithm is also included in these experiments. All three algorithms are coded using the toolkit of network modeling (TNM), a C++ class library specialized for modeling transportation networks (Nie, 2006). For the purpose of comparison, our TAPAS and iTAPAS programs are designed to share almost all components except for those related to the PAS identification and flow shift operations. All numerical results reported in this section were produced on a Windows 10 64-bit workstation with Intel Xeon CPU E3-1225 V3 3.2 GHz CPU and 12G RAM.

The algorithms are applied to four test networks, which are all available in http://www.bgu.ac.il/~bargera/tntp/. The properties of these networks are given in Table 1. The main convergence indicator used in this study is the relative gap (RG), which measures how close the current solution is to the precise UE solution. In our experiments, RG is calculated as,

$$\mathrm{RG} = 1 - \frac{\sum_{rs} u_{rs} q_{rs}}{\sum_{(i,j)\in A} x_{ij} t_{ij}} \tag{23}$$

where u_{rs} is the minimum travel cost between O-D pair r-s based on the current link travel cost t_{ij}, and q_{rs} is the travel demand between the O-D pair r-s. In all experiments, the convergence criterion is set to 10^{-13}. Finally, the Bureau of Public Roads (BPR) function is used to calculate link travel costs.

[5] In practical applications, some nodes will not be used at all by certain origins, and those links heading to these nodes will not be used at all. Some connectors will not be used as well. These links should be excluded when measuring the consistency.

Table 1
Test networks used in the numerical experiments.

Network	Number of nodes	Number of links	Number of zones	Number of O-D pairs	V/C ratio
Chicago Sketch	933	2950	387	142,890	0.54
PRISM	14,639	33,937	898	453,089	0.26
Chicago Regional	12,982	39,018	1771	3,136,441	0.60
Philadelphia	13,389	40,003	1525	1,150,722	0.75

Table 2
Computation times in terms of different convergence levels.

Network	Algorithm	Relative gap					
		10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-12}
Chicago sketch	FW	0.65 s	2.7 s	61 s			
	TAPAS	0.8 s	1.83 s	2.5 s	2.8 s	2.8 s	3.2 s
	iTAPAS	0.7 s	1.1 s	1.4 s	2.0 s	2.2 s	2.4 s
PRISM	FW	0.6 m	1.7 m	50 m			
	TAPAS	0.5 m	1.3 m	2.1 m	2.7 m	3.1 m	3.7 m
	iTAPAS	0.6 m	1.2 m	1.8 m	2.4 m	2.7 m	2.7 m
Chicago regional	FW	3.5 m	69 m				
	TAPAS	3.6 m	10.1 m	12.0 m	18.5 m	25.4 m	36.3 m
	iTAPAS	2.8 m	5.1 m	6.8 m	8.2 m	10.8 m	13.4 m
Philadelphia	FW	3.1 m	124 m				
	TAPAS	6.8 m	11.2 m	14.9 m	23.5 m	39.4 m	63.2 m
	iTAPAS	4.4 m	7.6 m	9.3 m	12.5 m	15.6 m	19.4 m

Note: *s*, seconds; *m*, minutes.

7.1. UE convergence analysis

This section compares the convergence performance of TAPAS and iTAPAS on the aforementioned networks with different sizes and congestion levels. Table 2 reports the computation times required by each algorithm to reach various levels of precision. Note that the proportionality operation step, Step 3.3 in Procedure 1 is removed in these results to make a fair comparison on their convergence speed, because the proportionality operation usually requires extra computation time (see Section 7.3). Missing values indicate that those levels of precision cannot be reached within two hours in all cases. As shown in the table, FW failed to reach RG $< 10^{-5}$ in almost all experiments—a widely known and notorious convergence behavior. In contrast, TAPAS and iTAPAS were able to reach RG $= 10^{-12}$ in all cases.

iTAPAS outperforms TAPAS in all of our experiments. Moreover, Table 2 shows that the relative performance of TAPAS and iTAPAS is significantly affected by network size and congestion level (where the latter is indicated by average volume/capacity (V/C) ratio). In general, the two algorithms tend to behave similarly in solving smaller and less congested networks, and the advantage of iTAPAS over TAPAS is much more prominent for larger and more congested networks, as demonstrated clearly in Fig. 5. For the Chicago Regional and Philadelphia networks, iTAPAS is at least twice as fast as TAPAS to achieve RG of 10^{-12}.

Recall that iTAPAS differs from TAPAS mainly in two algorithmic parts: The PAS identification method and the number of associated origins with each PAS. The results in the previous sections show the blended improvements from these two parts, which makes it hard to see which part plays a more significant role. To make it clearer, we implemented two more variants of TAPAS: One is the original TAPAS with a modification of only employing the new PAS identification method, denoted by TAPAS-n; the other is the original TAPAS with a modification of only associating single origin with each PAS, denoted by TAPAS-s. Their performance in solving the Chicago Regional and Philadelphia networks are plotted in Fig. 6. The results show that both algorithmic parts can significantly reduce the convergence time of TAPAS. More specifically, it seems to be safe to conclude that the improvement in the convergence efficiency of iTAPAS should attribute more to the new PAS identification method in the early stage of convergence, where the number of PASs is usually very small; and attribute more to the single-origin association mechanism in the latter stage of convergence when the size of the PAS list grows to be very large. Fig. 6 also justifies that the cross-origin adjustment does not actually help the UE convergence.

7.2. PAS-related statistical analysis

In this section, we further examine some statistical facts of PASs in TAPAS and iTAPAS, in order to provide possible explanations for their different performance.

A set of PAS statistics for each test network are reported in Table 3, which are based on the equilibrium results obtained for RG $= 10^{-12}$. First, the number of PASs maintained by iTAPAS is less than that by TAPAS. The possible reason is that PAS in iTAPAS is more likely to be eliminated before it is equilibrated, because the flow shift in iTAPAS is operated in single origin until the origin-specific flow on any segment reduces to zero, which is different from the mechanism in TAPAS, where the

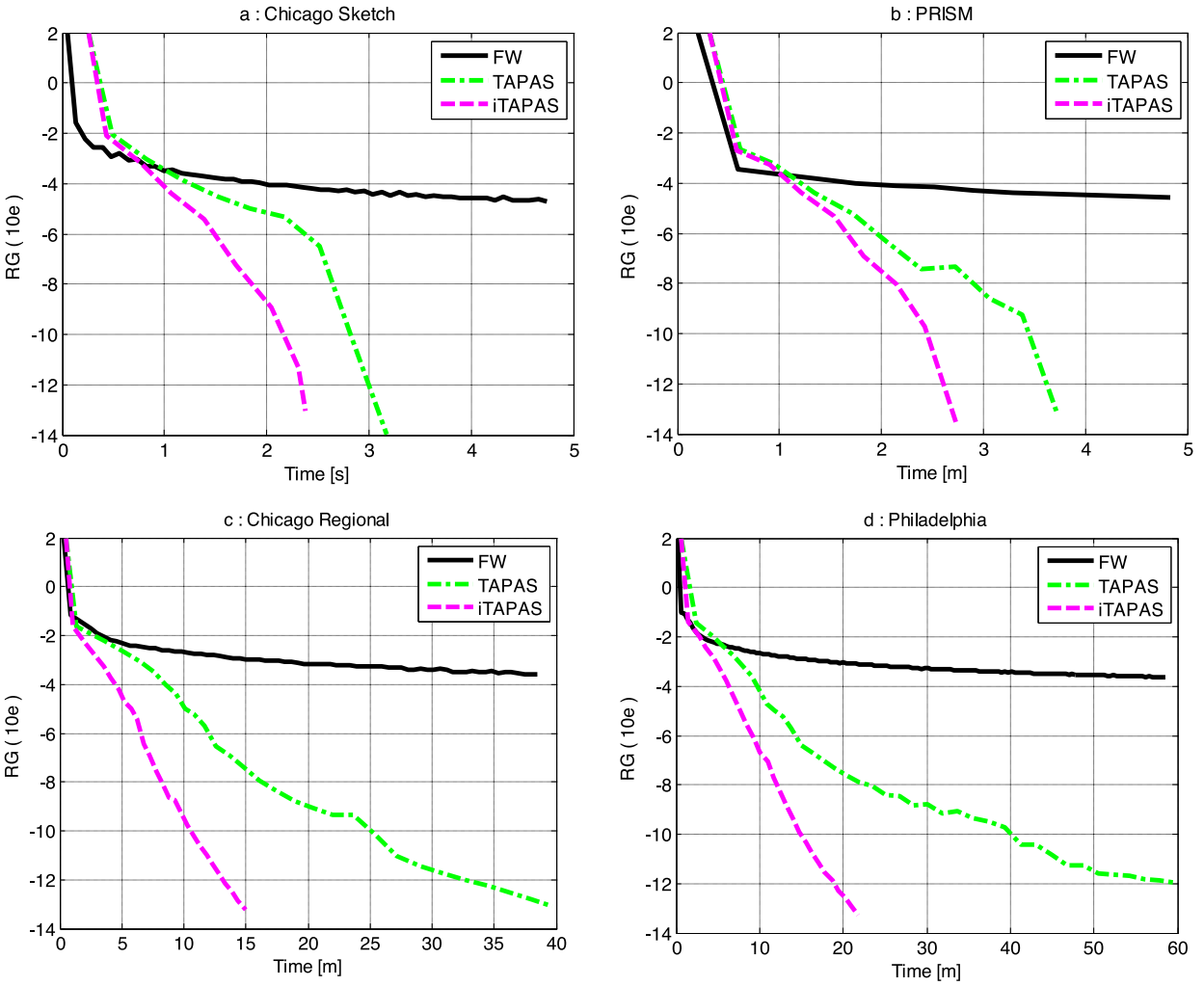

Fig. 5. Convergence performance on the test networks.

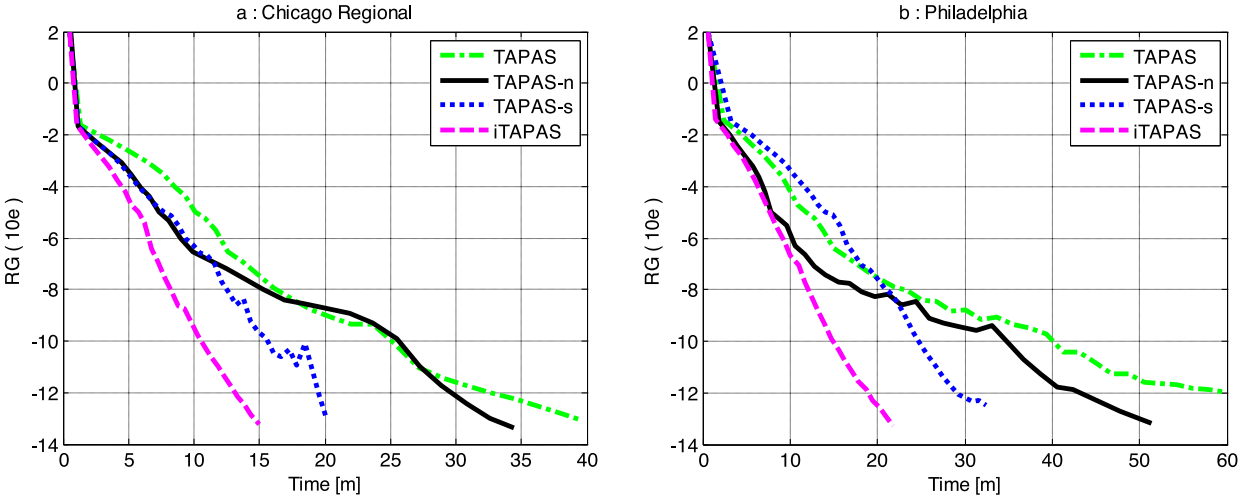

Fig. 6. Convergence performance of two variants of TAPAS.

Table 3
PAS set information for different test networks.

Network	Algorithm	Number of maintained PASs	Number of origins per PAS	Number of PAS shifts	Number of links per segment	Average f_e^r/x_{ij}^r ratio
Chicago sketch	TAPAS	329	2	47,416	4	0.102
	iTAPAS	313	1	56,242	3	0.146
PRISM	TAPAS	365	2	26,936	43	0.011
	iTAPAS	362	1	10,312	9	0.052
Chicago regional	TAPAS	20,860	41	1090,514	16	0.030
	iTAPAS	16,219	1	1045,823	7	0.065
Philadelphia	TAPAS	42,887	23	1242,400	19	0.024
	iTAPAS	27,370	1	1168,763	11	0.042

Table 4
Consistency and proportionality results produced by TAPAS and iTAPAS.

Algorithm	Post-process	Non-consistency condition 1	Non-consistency condition 2	Entropy value	Proportionality gap	Number of PASs	Computation time for post-process
TAPAS	Before	0	112,006	925,161.49	0.14	23,657	54.48 m
	After	0	1	951,016.78	1.31×10^{-6}	23,770	
iTAPAS	Before	0	916,718	449,148.03	0.57	22,139	101.17 m
	After	0	20	950,991.24	1.23×10^{-6}	25,326	

Note: m, minutes.

flow are shifted proportionally among all associated origins. Second, in TAPAS, the average number of origins associated with each PAS in the two large regional networks are much larger than one. In the Chicago Regional network, the average number of origins is as large as 41, and the additional computational efforts caused by the 41 origins can be intuitively illustrated by providing a conceptual computation comparison for the two algorithms in shifting PAS flow. If we define a flow addition/deduction operation for one segment as an elementary operation, then TAPAS needs $41 \times 2 \times 1{,}090{,}514 \approx 8.94 \times 107$ operations to shift PAS flows in each iteration; in comparison, iTAPAS only takes $1 \times 2 \times 1{,}045{,}823 \approx 2.09 \times 106$ operations, roughly 2.3% of that of TAPAS. Third, the number of PASs maintained in solving the Chicago Sketch and PRISM networks is much less than that in solving the Chicago Regional and Philadelphia networks for both algorithms. It seems that the more congested a network is, the more PASs the algorithms need. Thus, a more improved efficiency can be achieved by iTAPAS than TAPAS. Essentially, the above three observations not only show the advantage of associating only one origin to each PAS in the convergence efficiency, but also explain the fact that the performance of iTAPAS is comparable to that of TAPAS in solving smaller and less congested networks, while the former is significantly better than the latter for the two large-scale networks.

The sixth column shows the average number of links that a single segment includes over all PASs maintained by the algorithm; the seventh column is the average ratio of segment flow bound f_e^r on the higher cost segment to the origin-specific link flow on the potential link (i, j) for all newly identified PASs. This ratio reflects the average flow effectiveness of newly identified PASs. The results show that the proposed PAS identification method embedded in iTAPAS searches a smaller number of links in creating PASs and identifies those segments with a higher flow effective ratio. These observations confirm that the proposed PAS identification method is more efficient.

7.3. Consistency and proportionality

In Section 5, we sketched a post-process procedure to achieve the consistency and proportionality for iTAPAS. Now several statistical measures of the computational results are presented in the following to verify how much the two conditions are really satisfied after the procedures are executed.

The comparison result of consistency and proportionality for both TAPAS and iTAPAS in solving the Chicago Regional network is presented in Table 4. The fourth column shows the number of links meeting the condition in (17) (named the 1st non-consistency condition) and the fifth column shows the number of links meeting the condition in (18) (named the 2nd non-consistency condition). Note that obtaining perfect UE solutions for problems of realistic size is almost impossible due to the computational precision limit, so reduced cost precision errors smaller than 10^{-12} are all viewed as no cost difference here, i.e., $\pi_{ij}^r < 10^{-12}$ is practically regarded as $\pi_{ij}^r = 0$. Generally, the statistics in these two columns present the convergence level of consistency.[6] The numbers of links left in the 1st non-consistency condition by TAPAS and iTAPAS

[6] In measuring the consistency for the Chicago Regional network, 12,698,895 origin-based links in non-consistency condition 2 are excluded because their heading nodes are not used at all by the corresponding origins; 2,426,453 origin-based links are excluded because they are connectors not used by any origins or destinations; 4,834 origin-based links are excluded because they are not used by any origins.

are both zero, indicating that the solutions are highly converged to the equilibrium and no link with a reduced link cost larger than 10^{-12} carries positive flow. The number of links in the 2nd non-consistency condition exhibits some differences between the two algorithms: The number of links left by TAPAS in the 2nd non-consistency condition before the post-process is 112,006, much less than that by iTAPAS, 916,718; after the post-process is executed, the number of links in the 2nd non-consistency condition by TAPAS reduces to 1, and in contrast, the number by iTAPAS is reduced to 20. These results reflect the role of the proportionality step (Step 3.3 in Procedure 1) of TAPAS in helping spread flows to more links and nodes, some of which are not likely to be covered by the PASs collected in the post-process procedure. However, a difference of 19 links seems trivial compared to the total number of 916,718 links.

The sixth column shows the entropy value computed by the following origin-based formulation (Bar-Gera, 2010),

$$E = \sum_{r\in R}\sum_{(i,j)\in A} x_{ij}^r \cdot \ln\left(\frac{x_{ij}^r}{\eta_j^r}\right) \tag{24}$$

The seventh column shows the proportionality gap computed by

$$G_p = \frac{\sum_{p\in P}\max\{|\rho-\rho_r||r\in R_p\}}{|P|} \tag{25}$$

These two statistics can reflect the convergence of proportionality. For a specific network, we usually cannot exactly know the largest entropy value that can be achieved by the proportionality condition, but it is ensured that a larger entropy value represents a better convergence of proportionality. The entropy value of TAPAS before the post-process is significantly larger than that of iTAPAS, e.g., the difference is as large as 476,013.46; after the post-process, the difference of the entropy values between the two algorithms is only 25.54, which is almost negligible in practical applications. The proportionality gap G_p reflects the averaged convergence of the proportionality condition for a given set of PASs. Note that a smaller G_p for a given set of PASs does not necessarily mean a larger entropy value for the corresponding network, where the latter also depends on the consistency of the PAS set. For the given set of PASs, the post-process can converge to an average proportionality gap as small as $G_p = 10^{-7}$ within a reasonable time frame. Moreover, we observe that the post-process in TAPAS takes only about half of the time in iTAPAS to converge to the same precision level, due to the warm start in proportionality of the TAPAS solution. It is worth noting that extra computational time is required (compared to the computation time shown in Table 2) for TAPAS in the UE-finding process if the inter-proportionality step is activated, i.e., the extra time of about 48 minutes is needed in solving the Chicago Regional network. From the perspective of computing time, it seems that the consistency and proportionality operations will cost a comparable amount of time, no matter the operations are executed in the UE-finding process or in the post-process.

Basically, the above computational analysis justified the effectiveness of the post-process in achieving consistency and proportionality, and proved that the difference of the convergence levels between iTAPAS and TAPAS is negligible. As the total computational time consumed for consistency and proportionality is roughly equal, one advantage of performing consistency and proportionality in the post-process is that one can run the post-process only when it is required, such that considerable computation time can be saved in the UE convergence stage.

8. Concluding remarks

This paper carries out an algorithmic analysis and computational study on the use of PAS for TAP to a higher level than previous research. We presented several alternative conditions that help construct the relationship between UE flow patterns to PAS structures. These conditions explain well the motivation of this research and the advantages of PAS-based algorithms. A thorough review of TAPAS provides helpful insights, based on which an improved version of TAPAS where we name it iTAPAS is presented. Two main key improvements, a new PAS identification method and an approach of associating only one unique origin with each PAS, are proposed in iTAPAS to improve the efficiency of PAS operations. Numerical experiment results from applying iTAPAS for solving large networks show that this improved algorithm significantly increases the convergence speed. Several further comparisons of PAS set statistics used in testing TAPAS and iTAPAS, i.e., the number of PASs maintained in the solution process and the number of PAS flow shift operations, strongly support our observations.

PAS can also be used to achieve a path flow solution that meets consistency and proportionality without enumerating any UE paths. A post-process procedure is developed for iTAPAS (or other origin-based algorithms) to achieve the conditions of consistency and proportionality. Given a UE solution with a relative high convergence precision, the post-process redistributes flow for each potential origin-based link among origins by collecting PASs and origins associated with these PASs. Our numerical results show that the post-process can be used in iTAPAS to obtain a very close convergence level of consistency and proportionality to TAPAS, where the latter performs proportionality operations in both the UE-finding process and the post-process. To this end, placing the proportionality operation in the post-process is often preferred.

To date, the methodology and techniques of using PAS for achieving the UE convergence seems to be well established. However, a further utilization of PAS for proportionality realization and entropy maximization still remains as a subject to be explored in future research. A challenging and important task for this purpose is to formally establish the relationship between entropy maximization and proportionality realization. Meanwhile, developing origin-based algorithms for the entropy

maximization-based traffic assignment problem poses another interesting research topic that may benefit from exploiting the algorithmic potential of PAS.

Acknowledgments

We would like to thank Prof. Yu (Marco) Nie and Prof. David Boyce at Northwestern University for their helpful comments on earlier versions of this paper. Specifically, Prof. Nie suggested us to add Section 5 and Fig. 3 in this paper. This research is jointly sponsored by the National Nature Science Foundation of China (Grant No. 71501129, 71471111), the China Recruitment Program for Global Experts, the China Postdoctoral Science Foundation (Grant No. 2014M551414), and the Key State Laboratory of Ocean Engineering at Shanghai Jiao Tong University (Grant No. GKZD010061).

Appendix A

Proof for Proposition 1: Necessity. When a feasible solution of the problem in (1)–(3) is optimal, the conditions in (10) and (11) are satisfied for each origin. Without loss of generality, we choose any one segment e from the set E for the following analysis. According to Definition 3, $\pi_e^r = \sum_{(i,j)\in e} \pi_{ij}^r \geq 0$ is always true, the condition in (14) is justified. When $f_e^r = 0$, $f_e^r \pi_e^r = 0$ is true; when $f_e^r > 0$, which means that $x_{ij}^r > 0$ for all links $(i, j) \in e$ according to the definition of f_e^r, implying that $\pi_{ij}^r = 0$ for all links $(i, j) \in e$ according to the condition in (10), and hence we have $\pi_e^r = 0$. So the condition in (13) is proved.

Sufficiency. Let us restrict the size of segments to be one, and define the set of them as E^1. Then we have $E^1 := A$, and $E^1 \subseteq E$. If the conditions in (13) and (14) are satisfied, the following conditions must be satisfied as well,

$$f_e^r \pi_e^r = 0 \quad \forall e \in E^1, \ \forall r \in R$$

$$\pi_e^r \geq 0 \quad \forall e \in E^1, \ \forall r \in R$$

which are equivalent to the conditions in (10) and (11) for each origin according to Definition 3. □

Proof for Corollary 1. Consider a segment e that consists of link (i, j). If $x_{ij}^r > 0$ and $\pi_{ij}^r > 0$, then $\pi_e^r > 0$ is true according to Definition 3. Further, given $f_e^r > 0$, then $f_e^r \pi_e^r > 0$ is true. □

Proof for Proposition 2. Sufficiency. All links can be classified into three categories: For any link (i, j) that is used by origin r while being not on the shortest path tree, if there is at least such one equilibrated PAS segment $p(e_1, e_2)$ that e_1 is the shortest path segment and e_2 includes link (i, j), we have $\pi_{e_2}^r = t_{e_2} - t_{e_1} = 0$. Considering that π_{ij}^r is non-negative and $\pi_{e_2}^r = \sum_{(i,j)\in e_2} \pi_{ij}^r$, we must have $\pi_{ij}^r = 0$, or $x_{ij}^r \pi_{ij}^r = 0$; for any link (i, j) that is not used by origin r while being no on the tree, $x_{ij}^r \pi_{ij}^r = 0$ is true because $x_{ij}^r = 0$; for any link (i, j) that is on the shortest path tree, $\pi_{ij}^r = 0$ is obvious. Hence, all links satisfy the optimality conditions based on link reduced cost.

Necessity. If a feasible solution is optimal, then it satisfies the optimality conditions based on link reduced cost. Given a used link (i, j) that is also not on the shortest path tree rooted at r, we can always find such an effective PAS $p(e_1, e_2)$ for (i, j) by the PAS search method, when e_1 is the shortest path segment and e_2 carries flow larger than zero and includes (i, j). Further, all used links have a reduced cost equal to zero, i.e., $\pi_{e_2}^r = 0$ or $t_{e_2} = t_{e_1}$. Hence, p is an equilibrated PAS. □

Appendix B

In Step 1.1 of Procedure 5, directed cycles are detected and removed by a depth first search (DFS) method realized in a recursive manner. If we name this method as DFSCycle(i), where the argument i denotes a node, then the cycle-removing procedures can be descripted as follows.

Input: Initialize node i as the origin r; set a scan status for each node $l_k = 0, \ \forall k \in N$. Set a node set $Q = \emptyset$. Then define DFSCycle(i):

Step 1: If $l_i = 1$, a cycle is found ending at i, do the following:
- *1.1* Set a cycle node set $C = \emptyset$, put node i in to C from the back;
- *1.2* Collect each node k in set Q from the back, until $k = i$, and push these nodes in to set C orderly.
- *1.3* Define cycle link set D according to set C, and find out the minimum flow of cycle link flow $dx = \min\{x_{ij}^r | (i, j) \in D\}$. Update $x_{ij} = x_{ij} - dx$ and $x_{ij}^r = x_{ij}^r - dx$, update the travel time t_{ij} and its derivative t_{ij}'. Return.

Step 2: Else if $l_i = 0$, do the following:
- *2.1* Set $l_i = 1$, push node i in to set Q from the back.
- *2.2* For every link (i, j) in the forward link set of node i, set $i = j$ and invoke DFSCycle(i).
- *2.3* Take out the last node of set Q from the back.

Procedure 5 iTAPAS algorithm.

Input: Parameters ϵ, θ, μ, ν

Step 0: For each origin $r \in N$, initialize the subnetwork G_r as the shortest path tree rooted at r. Assign all the flow to links in the tree, and update link travel times t_{ij} and their derivatives t'_{ij}. Create a set $P = \emptyset$ to keep PASs over the network, a set $P_j = \emptyset$ for each node $j \in N$ to keep PASs, the head of which is j, and a link set $L_r = \emptyset$ for each origin to keep the potential links that violate the optimality conditions. A PAS is denoted by $p(e_1, e_2)$, and the beginning and ending nodes of it are denoted by p_t and p_h, respectively.

Step 1: For each origin r, do the following:

1.1 Update the shortest path tree T_r rooted at r. Calculating the reduced cost π^r_{ij} for each non-tree link $(i, j) \in G_r \backslash T_r$, and if $x^r_{ij} > \epsilon$ and $\pi^r_{ij} > \theta$, push (i, j) in to L_r.

1.2 While $L_r \neq \emptyset$, take a potential link (i, j) out of L_r, and do the following steps:

1.2.1 PAS match: Search each PAS $p(e_1, e_2) \in P$. If there is one PAS satisfying the following conditions, shift flow for this PAS at its associated origin r_0, and repeat 1.2 if $f^{r_0}_{e_2} > \epsilon$ after the flow shift; otherwise, go to 1.2.2. The conditions are:

(1) $t_{e_2} - t_{e_1} > \mu\pi^r_{ij}$;

(2) $f^{r_0}_{e_2} > \nu x^r_{ij}$;

(3) $(i, j) \in e_2$.

1.2.2 PAS identification: Identify a new PAS $p(e_1, e_2)$ for link (i, j) by the MFS method.

1.2.3 Check if $p(e_1, e_2)$ already exists in P_j (where it does not matter which origin relates to this PAS). If not, associate the current origin r to $p(e_1, e_2)$, and push it into P_j and P; otherwise, eliminate it.

1.3 Local PAS flow shift: Randomly choose a small number of PASs (i.e., 100 for small networks and 400 for large networks) from P and do flow shift for them to fasten the overall convergence of algorithm.

Step 2: Repeat the following steps for 20 times.

For each PAS $p(e_1, e_2) \in P$, do the following:

2.1 If the segment flow bound $f^{r_0}_{e_1} = 0$ or $f^{r_0}_{e_2} = 0$ on its associated origin r_0, and $t_{e_1} \neq t_{e_2}$, search a new origin for this PAS before we eliminate $p(e_1, e_2)$ from P and P_j: search a number of origins (i.e., $[r_0 + 1, \cdots, r_0 + 50]$); if $f^{r_0+i}_{e_2} > 0$, replace origin r_0 with $r_0 + i$ for this PAS and do the flow shift operation once; otherwise, go to Step 2.2.

2.2 Shift flow from the higher cost segment to the lower cost segment to equalize their cost.

Step 3: If the convergence criterion is achieved, go to Step 4; otherwise, go to Step 1.

Step 4: Do the post-process for consistency and proportionality by Procedure 6.

Appendix C

In summary, the post-process for consistency and proportionality is described as follows

Procedure 6 A post-process for consistency and proportionality.

Input: An origin-specific link flow UE solution, a PAS set P collected in the UE-finding process, and a predefined criterion ε.

Step 0: For each origin $r \in R$, calculate the origin-based node flow η^r_j and link approach proportion ϕ^r_{ij}.

Step 1: For each origin $r \in R$, if link (i, j) satisfies the condition in (18) at origin r, identify a PAS $p(e_1, e_2)$ that segment 1 incorporates (i, j) and segment 2 obeys $g^r_{e_2} > 0$. Add the identified PAS into the set P.

Step 2: For each PAS $p(e_1, e_2) \in P$, while the proportion differences between origins are not within the accepted range, do the following:

2.0 Collect all origins associated to this PAS and keep them in the set R_p.

2.1 For each origin $r \in R_p$: Calculate the segment flow bound $f^r_{e_1}$ and $f^r_{e_2}$ by (12); calculate the segment flow $g^r_{e_1}$ and $g^r_{e_2}$ by (19); calculate the origin-specific proportion ρ^r by (20).

2.2 Calculate the PAS proportion ρ by (21). For each origin $r \in R_p$, compute the flow adjustment δ_r by (22). Set

$$x^r_{ij} = \begin{cases} x^r_{ij} + \delta_r \ \forall (i, j) \in e_1 \\ x^r_{ij} - \delta_r \ \forall (i, j) \in e_2 \end{cases}$$

2.3 Update the η^r_j and ϕ^r_{ij} for all involved origins, nodes and links.

Step 3: Proportionality convergence test: Compute the average proportion gap as $G_p = \sum_{p \in P} \max\{|\rho - \rho_r|, \ \forall r \in R_p\}$, if $G_p < \varepsilon$, stop; otherwise, go to Step 1.

References

Bar-Gera, H., 2002. Origin-based algorithm for the traffic assignment problem. Transp. Sci. 36, 398–417.

Bar-Gera, H., 2006. Primal method for determining the most likely route flows in large road networks. Transp. Sci. 40, 269–286.

Bar-Gera, H., 2010. Traffic assignment by paired alternative segments. Transp. Res. Part B Methodological 44, 1022–1046.

Bar-Gera, H., Boyce, D., 1999. Route flow entropy maximization in origin-based traffic assignment. In: Proceedings of the 14th International Symposium on Transportation and Traffic Theory. Jerusalem, Israel, pp. 397–415.

Bar-Gera, H., Boyce, D., Nie, Y.M., 2012. User-equilibrium route flows and the condition of proportionality. Transp. Res. Part B 46, 440–462.

Beckmann, M., McGuire, C., Winsten, C.B., 1956. Studies in the Economics of Transportation. Yale University Press, New Heaven, CT.

Bell, M.G., Iida, Y., 1997. Transportation Network Analysis. John Wiley & Sons, West Sussex, England, UK.

Boyce, D., Xie, J., 2013. Assigning user class link flows uniquely. Transp. Res. Part A Policy and Practice 53, 22–35.

Dial, R.B., 1971. A probabilitstic multipath traffic assignment algorithm which obviates path enumeration. Transp. Res. 5, 83–111.

Dial, R., 1999. Accurate Traffic Equilibrium: How to Bobtail Frank-Wolfe. Volpe National Transportation Research Center, Cambridge, MA Technical Report.

Dial, R.B., 2006. A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration. Transp. Res. Part B Methodological 40, 917–936.

Florian, M., Constantin, I., Florian, D., 2009. A new look at projected gradient method for equilibrium assignment. Transp. Res. Rec. 2090, 10–16.

Florian, M., Guálat, J., Spiess, H., 1987. An efficient implementation of the "PARTAN" variant of the linear approximation method for the network equilibrium problem. Networks 17, 319–339.

Fukushima, M., 1984. A modified Frank-Wolfe algorithm for solving the traffic assignment problem. Transp. Res. Part B Methodological 18, 169–177.

Gentile, G., 2012. Local user cost equilibrium: a bush-based algorithm for traffic assignment. Transportmetrica 10, 1–40.
Inoue, S., Maruyama, T., 2012. Computational experience on advanced algorithms for user equilibrium traffic assignment problem and its convergence error. In: Proceedings of the 8th International Conference on Traffic and Transportation Studies. Changsha, China, pp. 445–456.
Jayakrishnan, R., Tsai, W.T., Prashker, J.N., Rajadhyaksha, S., 1994. Faster path-based algorithm for traffic assignment. Transp. Res. Rec. 1443, 75–83.
Kumar, A., Peeta, S., 2011. An improved social pressure algorithm for static deterministic user equilibrium traffic assignment problem. In: Proceedings of the Transportation Research Board 90th Annual Meeting. Washington, DC.
Kumar, A., Peeta, S., 2015. Entropy weighted average method for the determination of a single representative path flow solution for the static user equilibrium traffic assignment problem. Transp. Res. Part B Methodological 71, 213–229.
Kumar, A., Peeta, S., Nie, Y., 2012. Update strategies for restricted master problems for user equilibrium traffic assignment problem: computational study. Transp. Res. Rec. 2283, 131–142.
Larsson, T., Patriksson, M., 1992. Simplicial decomposition with disaggregated representation for the traffic assignment problem. Transp. Sci. 26, 4–17.
LeBlanc, L.J., Morlok, E.K., Pierskalla, W.P., 1975. An efficient approach to solving the road network equilibrium traffic assignment problem. Transp. Res. 9, 309–318.
Lee, D.-H., Nie, Y., Chen, A., 2003. A conjugate gradient projection algorithm for the traffic assignment problem. Math. Comput. Model. 37, 863–878.
Nguyen, S., 1974. An algorithm for the traffic assignment problem. Transp. Sci. 8, 203–216.
Nie, Y., 2006. A Programmer's Manual for Toolkit of Network Modeling (TNM). University of California, Davis, CA.
Nie, Y.M., 2010. A class of bush-based algorithms for the traffic assignment problem. Transp. Res. Part B Methodological 44, 73–89.
Nie, Y.M., 2012. A note on Bar-Gera's algorithm for the origin-based traffic assignment problem. Transp. Sci. 46, 27–38.
Rossi, T.F., McNeil, S., Hendrickson, C., 1989. Entropy model for consistent impact-fee assessment. J. Urban Plann. Dev. 115, 51–63.
Wardrop, J., 1952. Some theoretical aspects of road traffic research. In: Proceedings of the Institution of Civil Engineers Part II, 1, pp. 325–378.
Xie, J., Nie, Y.M., Yang, X., 2013. Quadratic approximation and convergence of some bush-based algorithms for the traffic assignment problem. Transp. Res. Part B Methodological 56, 15–30.
Xie, J., Xie, C., 2015. Origin-based algorithms for traffic assignment: Algorithmic structure, complexity analysis, and convergence performance. Transp. Res. Rec. 2498, 46–55.
Zheng, H., 2015. Adaptation of network simplex for the traffic assignment problem. Transp. Sci. 49, 543–558.