Cost Scaling Based Successive Approximation Algorithm for the Traffic Assignment Problem

Hong Zheng[a], Srinivas Peeta[b*]

a. NEXTRANS Center, Purdue University, 3000 Kent Ave., West Lafayette, IN 47906, USA
b School of Civil Engineering, Purdue University, 550 Stadium Mall Drive, West Lafayette, IN 47907, USA

*Corresponding author. Tel.: +1 765 494 2209; fax: +1 765 496 7996.
E-mail addresses: zheng225@purdue.edu (H. Zheng), peeta@purdue.edu (S. Peeta)

Abstract

This paper presents a cost scaling based successive approximation algorithm, called ε-BA (ε-optimal bush algorithm), to solve the user equilibrium traffic assignment problem by successively refining ε-optimal flows. As ε reduces to zero, the user equilibrium solution is reached. The proposed method is a variant of bush-based algorithms, and also a variant of the min-mean cycle algorithm to solve the min-cost flow by successive approximation. In ε-BA, the restricted master problem, implying traffic equilibration restricted on a bush, is solved to ε-optimality by cost scaling before bush reconstruction. We show that ε-BA can reduce the number of flow operations substantially in contrast to Dial's Algorithm B, as the former operates flows on a set of deliberately selected cycles whose mean values are sufficiently small. Further, the bushes can be constructed effectively even if the restricted master problem is not solved to a high level of convergence, by leveraging the ε-optimality condition. As a result, the algorithm can solve a highly precise solution with faster convergence on large-scale networks compared to our implementation of Dial's Algorithm B.

Key words: traffic assignment, successive approximation, cost scaling, min-mean cycle, bush

1. Introduction

The traffic assignment problem (TAP) lies at the core of transportation planning. The user-equilibrium (UE) TAP has had widespread applications in the last five decades since Beckmann formulated it as a mathematical programming problem (Beckmann et al., 1956). The basic model continues to be widely used, and finding fast algorithms to solve the TAP is of sustained interest to both researchers and practitioners world-wide.

Algorithms to solve the TAP typically operate in the space of link flows (Frank and Wolfe, 1956; LeBlanc et al., 1975), and route flows (Dafermos, 1968; Florian et al., 2009; Jayakrishnan et al., 1994; Larsson and Partriksson, 1992). It has long been known that the TAP is equivalent to a min-cost multi-commodity flow problem on an uncapacitated network with a convex, continuously differentiable objective function, where each commodity is characterized as originating from a single origin. Thereby, algorithms (Nguyen, 1974; Petersen, 1975) have been proposed to decompose the master problem, by solving the single commodity subproblems repeatedly until all of the commodities are optimized simultaneously. Such algorithms have been characterized in recent years as "origin-based".

Origin-based algorithms have recently garnered significant focus in the TAP context due to a family of bush-based algorithms (Bar-Gera, 2002; Dial, 2006; Nie, 2010; Nie, 2012). These algorithms maintain an acyclic subnetwork, labeled a bush (Dial, 2006), at each origin and restrict origin-based assignments only to those bushes. They repeatedly solve two subproblems: bush equilibration and bush reconstruction (Nie, 2010). The bush equilibration subroutine solves for the UE flow restricted on a bush, also called the restricted master problem (RMP). The acyclicity of a bush is a key building block in the design of a fast algorithm in this family, which allows the fastest possible shortest path tree and, perhaps more importantly, the longest path tree computations. Hence, the bush equilibration can be performed trivially by shifting flows between the longest and shortest paths (Dafermos, 1971; Dial, 2006). In our opinion, a formidable challenge for bush-based algorithms that may preclude very precise convergence (for example, at a relative gap of 10^{-12}) on large-scale networks is how to reconstruct bushes. Because of the requirement of acyclicity for a bush, a non-bush arc that violates optimality conditions may not be appropriately manipulated in bush reconstruction due to the likelihood of acyclicity violation (see Section 3.1). As a result, a bush algorithm may not approach convergence if the bush reconstruction subproblem does not converge to the optimal bush, unless a method is explicitly developed to resolve this issue.

In this paper we present an algorithm called ε-optimal[1] bush algorithm (ε-BA) to solve the TAP by successive approximation. The method is inspired by the min-mean cycle method (Goldberg and Tarjan, 1989) and the successive approximation algorithms for solving the min-cost flow (MCF) problem (Bland and Jensen, 1992; Goldberg and Tarjan, 1990; Rock, 1980). It modifies the RMP in a bush algorithm by operating flows only on a set of selected cycles[2] whose negative mean-costs are small, rather than all negative cycles. A min-mean cycle can be found in $O(|N||A|)$ time (Karp, 1978), which can be fairly expensive. Hence, a cost scaling method is applied to identify a set of cycles with negative mean-costs at most as a predefined threshold value $-\lambda$. The essence of ε-BA is to combine two components: (1) to operate flows on a set of selected cycles whose mean-costs are negatively small; and (2) to solve the set of selected cycles by leveraging the acyclicity of a bush. The proposed algorithm is a variant of bush-based algorithms, and also a variant of the min-mean cycle algorithm. We show that ε-BA could lead to a significantly reduced number of flow operations. In ε-BA, we solve the RMP to ε-optimality before proceeding to the bush reconstruction, and successively refine ε-optimality by reducing ε. As $\varepsilon \downarrow 0$ (ε reduces to zero), ε-BA reduces to a bush algorithm as in Dial (2006) or Nie (2010). We show that ε-BA has the following merits compared to a bush algorithm: (i) ε-BA provides a benchmark indicating when to stop solving the RMP while the RMP is not solved precisely; (ii) given ε-optimality of a bush, ε-BA provides a new rule to reconstruct bushes warranting acyclicity while not necessarily solving the RMP very precisely; (iii) as the key finding of the study, the computational effort of ε-BA can be significantly less than that for a bush algorithm; and (iv) ε-BA can produce a highly precise solution with faster convergence than a bush algorithm, especially on large-scale networks.

The remainder of the paper is organized as follows. Section 2 provides some preliminaries in terms of notation and definitions for the problem. Section 3 provides an overview of the ε-BA. Section 4 presents the cost scaling method to refine ε-optimality restricted on a bush. Section 5 discusses several implementation issues. Section 6 presents numerical experiments to demonstrate the computational performance and convergence properties of the proposed algorithm. Finally, some concluding comments are provided in Section 7.

2. Preliminaries

2.1 Notation

A transportation network $G = (N, A)$ consists of a set of nodes N and a set of directed arcs A (also called links). Suppose G is strongly connected, that is, at least one route exists between any two nodes in G. Flow assigned on arc $(i,j) \in A$ is denoted by x_{ij}. Each arc $(i,j) \in A$ is associated with an unbounded capacity denoted by u_{ij}, and a nonnegative flow-dependent cost

[1] We note that Dial (2006) also mentioned ϵ-UE. Dial's ϵ-UE implies that the travel times on all paths are within ϵ of the cheapest path (c.f. Section 6). Our ε-optimality condition, however, is in the context of the reduced costs specified in Eq. (6), which is closely related to the min-mean cycle in the residual network.

[2] Flow operations in this paper are manipulated on a set of cycles called PAS - Paired Alternative Segments. See Definition 1.

(or travel time) $c_{ij}(x_{ij})$, or simply c_{ij} for notational convenience. N consists of a set of origins denoted by $\mathcal{P}$, a set of destinations denoted by $\mathcal{Q}$, and a set of intermediate nodes. A set of origin-destination (O-D) pairs is denoted by a vector $V \coloneqq \{(p,q) \mid p \in \mathcal{P}, q \in \mathcal{Q}\}$. In the origin-based TAP, the set of destinations associated with an origin $p \in \mathcal{P}$ is denoted by $\mathcal{Q}(p)$. Demand from each origin $p \in \mathcal{P}$ to its destination $q \in \mathcal{Q}(p)$ is denoted by b_{pq}. Denote the total demand of origin p by B_p, where $B_p = \sum_{q \in \mathcal{Q}(p)} b_{pq}$. Denote by $I(i)$ the set of inbound arcs ending at node i, and $O(i)$ the set of outbound arcs incident from node i.

A walk visits a list of nodes $i_1, \dots, i_n$ such that for each $k = 1, \dots, n-1$ there is $(i_k, i_{k+1}) \in A$ or $(i_{k+1}, i_k) \in A$. The walk is directed if $(i_k, i_{k+1}) \in A$ for each $k = 1, \dots, n-1$. A cycle, denoted by W, is a walk where $i_1 = i_n$ and $n > 2$. A directed cycle is a directed walk with the same condition of a cycle. $|W|$ denotes the length (cardinality) of W, or the number of arcs contained in W. A subgraph is a graph (N', A') such that $N' \subseteq N$ and $A' \subseteq A$. In this paper, an arc (i,j) with $x_{ij} > 0$ is called an active arc, and with $x_{ij} = 0$ is called an inactive arc.

A cycle W has an orientation (also called direction), which is either clockwise or counter-clockwise. The set of arcs in W that have the same direction as the cycle is called the set of forward arcs, denoted by $F(W)$. The set of arcs that has the reverse direction as that of the cycle is called the set of backward arcs, denoted by $B(W)$. The cost of a cycle is the algebraic sum of the costs of all arcs contained in the cycle, i.e., $c_W(\mathbf{x}) = \sum_{(i,j) \in F(W)} c_{ij}(x_{ij}) - \sum_{(i,j) \in B(W)} c_{ij}(x_{ij})$. The mean-cost of a cycle is denoted by $c_W(\mathbf{x})/|W|$. Given a cycle W, the set of nodes that have two outgoing arcs in W is denoted by $O(W)$; the set of nodes with two incoming arcs in W is denoted by $I(W)$. Given a set Y, $|Y|$ denotes the cardinality of Y.

Proposition 1. A cycle W has $|I(W)| \leq \frac{1}{2}|W|$ and $|O(W)| \leq \frac{1}{2}|W|$.

Proof. A node $k \in I(W)$ (or $k \in O(W)$) is associated with one forward arc and one backward arc exactly, and each arc is associated with at most one node in $I(W)$ (or $O(W)$). Suppose $|I(W)| > \frac{1}{2}|W|$ (or $|O(W)| > \frac{1}{2}|W|$); it implies the node set $I(W)$ (or $O(W)$) is associated with a number of arcs more than $|W|$, which is clearly impossible. □

Next, some definitions used in this paper are summarized.

Definition 1. (PAS (Bar-Gera, 2010)). A Paired Alternative Segments (PAS) is a cycle W such that $|I(W)| = |O(W)| = 1$. A PAS always consists of two segments (or two directed paths), where one is short and the other is long, from the node in $O(W)$ to the node in $I(W)$. A PAS is denoted by $PAS = \{e_s, e_l\}$, where e_s and e_l denote the set of arcs in the short and long segments, respectively. The direction of a PAS is in the same direction such that e_s is forward. Therefore, the cost of a PAS is nonpositive.

Definition 2. (bush). A directed subgraph is called a bush, denoted by $\mathcal{B} = (N_B, A_B)$, if there is no directed cycle in $\mathcal{B}$.

Definition 3. (child bush). A bush $\mathcal{B}' = (N'_B, A'_B)$ is called a child bush of $\mathcal{B} = (N_B, A_B)$ if $N'_B \subseteq N_B$ and $A'_B \subseteq A_B$.

Definition 4. (spanning bush). A bush $\mathcal{B} = (N_B, A_B)$ is called spanning if $N_B = N$.

Definition 5. (flow-spanned bush). A flow-spanned bush, denoted by $\mathcal{B}_f = (N', A')$, is a child bush of $\mathcal{B}$ such that $A' = \{(i,j) \mid x_{ij} > 0\} \cap \mathcal{B}$.

Definition 6. (residual network). The residual network of $G = (N, A)$ with respect to $\mathbf{x}$, denoted by $G(\mathbf{x})$, is a directed auxiliary network. For an arc $e = (i,j) \in A$, we denote the corresponding reverse arc (j,i) by $e^- \coloneqq (j,i)$ and $e^- \in A^-$. The cost of a reverse arc e^- with $e \in A$ is $c_{e^-} = -c_e$. Given a feasible flow $\mathbf{x}$ in G, the residual capacity of arc $e \in A$ is $u_e - x_e$, and the residual capacity of the corresponding reverse arc e^- is x_e. The residual network $G(\mathbf{x})$ consists of all arcs in $A \cup A^-$ with positive residual capacity.

Definition 7. (topological order). A node label $\mathbf{s} = (s_i)_{i\in N}$ is called a topological order if for each $(i,j) \in A$ there is $s_i < s_j$.

Definition 8. (descending pass). A descending pass is a sequential visit of nodes in a bush in the decreasing topological order.

2.2. Origin-based traffic assignment

The proposed successive approximation algorithm is origin-based. The origin-based traffic assignment formulation is briefly reviewed hereafter.

Decompose the flow vector $\mathbf{x} = (x_{ij})_{(i,j)\in A}$ to a set of commodities, that is, $\mathbf{x} = \sum_{p\in\mathcal{P}} \mathbf{x}^p$, where each commodity $\mathbf{x}^p = (x_{ij}^p)_{(i,j)\in A}$ represents the flow sourced from origin p. The TAP can be reformulated by using the origin-based flow as follows (Bar-Gera, 2002; Nie, 2010):

$$min\, Z = \sum_{(i,j)\in A} \int_0^{\mathbf{x}(B)+x_{ij}^p} c_{ij}(u)du \tag{1a}$$

s.t.

$$\sum_{(i,j)\in O(i)} x_{ij}^p - \sum_{(j,i)\in I(i)} x_{ji}^p = \rho_i^p \qquad \forall i \in N, p \in \mathcal{P} \tag{1b}$$

$$\rho_i^p = \begin{cases} \sum_{q\in\mathcal{Q}(p)} b_{pq}, & i = p \\ -b_{pq}, & i = q \\ 0 & otherwise \end{cases} \qquad \forall i \in N, p \in \mathcal{P} \tag{1c}$$

$$x_{ij}^p \geq 0 \qquad \forall (i,j) \in A, p \in \mathcal{P} \tag{1d}$$

$$\mathbf{x}(B) = \sum_{p'\in\mathcal{P}\setminus\{p\}} x_{ij}^{p'} \qquad \forall (i,j) \in A, p \in \mathcal{P} \tag{1e}$$

In the design of an algorithm that iterates among origins, $\mathbf{x}^p$ is the flow variable, and the flow originating from other origins, denoted by $\mathbf{x}(B)$, is the constant background flow. When $\mathbf{x}(B)$ is considered to be constant, Eqs. (1a)-(1e) could also be deemed as the TAP on a single origin network. For an origin-based flow formulated in (1), we say a flow vector that satisfies (1b)-(1d) is feasible.

For notational simplicity the superscript p is omitted in the discussions hereafter on origin-based flows. Let $\mathbf{x}$ be a flow vector, and $\boldsymbol{\gamma} = (\gamma_i \gtreqless 0)_{i\in N}$ be a dual vector associated with Eq.

(1b); then, the Kuhn-Tucker conditions of the problem (1) are (Patriksson, 1994):

$$c_{ij}(x_{ij}) + \gamma_i - \gamma_j \geq 0 \qquad \forall (i,j) \in A \qquad (2a)$$
$$x_{ij} \cdot [c_{ij}(x_{ij}) + \gamma_i - \gamma_j] = 0 \qquad \forall (i,j) \in A \qquad (2b)$$

Denote the reduced cost $\mathbf{c}^{\boldsymbol{\gamma}} = (c_{ij}^{\gamma})_{(i,j)\in A}$ by:

$$c_{ij}^{\gamma} = c_{ij}(x_{ij}) + \gamma_i - \gamma_j \qquad \forall (i,j) \in A \qquad (3)$$

where the superscript $\boldsymbol{\gamma}$ means the reduced cost is calculated with respect to the dual vector $\boldsymbol{\gamma}$; reduced cost vectors calculated with other dual vectors follow the same exposition hereafter. As the feasibility constraint $x_{ij} \geq 0$ holds for each link $(i,j) \in A$, the Kuhn-Tucker conditions become Eq.(4):

$$x_{ij} = 0 \Rightarrow c_{ij}^{\gamma} \geq 0 \qquad \forall (i,j) \in A \qquad (4a)$$
$$x_{ij} > 0 \Rightarrow c_{ij}^{\gamma} = 0 \qquad \forall (i,j) \in A \qquad (4b)$$

Let $\mathbf{x}^*$ be the optimal flow vector, and let $\boldsymbol{\gamma}^*$ be the optimal dual vector. If $\gamma_r^* = 0$, where r denotes the root, $\boldsymbol{\gamma}^*$ is then the shortest distance label calculated in G, denoted by $\mathbf{d}$ (Patriksson, 1994). To facilitate our technical discussions later, we replace $\boldsymbol{\gamma}^*$ by $\mathbf{d}$ and then have the following well-known reduced cost optimality condition.

$$x_{ij}^* = 0 \Rightarrow c_{ij}^{d} \geq 0 \qquad \forall (i,j) \in A \qquad (5a)$$
$$x_{ij}^* > 0 \Rightarrow c_{ij}^{d} = 0 \qquad \forall (i,j) \in A \qquad (5b)$$

Given a bush, if all non-bush arcs have nonnegative reduced costs, the bush is then called the UE bush.

Definition 9. (UE bush (Nie, 2010)) A bush is a UE bush, denoted by $\mathcal{B}_{UE} = (N_B, A_B)$, if $\{(i,j) \mid c_{ij}^{d} < 0\} \cap A \backslash A_B = \emptyset$.

3. Bush and ε-optimal bush algorithms

In this section, we provide a high-level overview of the overall algorithmic procedure of a bush algorithm, and the procedure of the ε-optimal bush algorithm proposed in this paper. For details of a bush algorithm, refer to Dial (2006) and Nie (2010); we assume readers are familiar with those algorithms.

3.1 Bush algorithm (BA) (Dial, 2006; Nie, 2010)

The steps of a bush algorithm (BA) are as follows:

BA:
Step 0: Initialize a spanning bush $\mathcal{B}$; build an initial feasible solution in $\mathcal{B}$.
Step 1: Reconstruct $\mathcal{B}$ by adding a set of arcs that has the potential to improve the objective function value.

Step 2: Solve for the equilibrium flow in $\mathcal{B}$ (RMP).
Step 3: Convergence test. If converged, stop; otherwise go to Step 1.

In a bush algorithm, Step 1 iteratively reconstructs bush $\mathcal{B}$ until it converges to a UE bush $\mathcal{B}_{UE}$, and Step 2 solves for the equilibrium flows restricted on $\mathcal{B}$. These two steps are solved repeatedly and iteratively until both are solved to optimality simultaneously. The key challenge of a bush algorithm is how to reconstruct $\mathcal{B}$ closer to $\mathcal{B}_{UE}$ in Step 1. The acyclicity of a bush must be maintained. Let $\mathbf{d}$ and $\mathbf{D}$ be the shortest and longest distance labels calculated in $\mathcal{B}$, respectively. The reduced cost optimality condition implies that the addition of the set of arcs $A_1 := \{(i,j) \mid c_{ij}^d < 0, (i,j) \notin \mathcal{B}\}$ can rebuild $\mathcal{B}$ closer to $\mathcal{B}_{UE}$, as suggested by Dial (2006). This rule, however, may lead to directed cycles unless the RMP is solved to a high level of convergence. Pursuing a highly converged RMP solution is expensive and may not be worthwhile, as the bush will be reconstructed over iterations; and once done, one needs to resolve the RMP again. Bar-Gera (2002) suggests another rule; to add the set of arcs $A_2 := \{(i,j) \mid c_{ij}^D < 0, (i,j) \notin \mathcal{B}\}$ to reconstruct bushes. A_2 guarantees acyclicity though it is not much effective in pursuing convergence as it does not guarantee $c_{ij}^d < 0$; and if so, (i,j) should not be added to the bush, at least at this point, due to the reduced cost optimality condition. To secure acyclicity given a loose RMP solution, Nie (2010) suggests a joint rule by adding the set of arcs $A_3 := A_1 \cap A_2$; in case $A_3 = \emptyset$ then add A_2 exclusively. In a large-scale network, it is not unusual to encounter $A_3 = \emptyset$, hence a bush algorithm may have difficulty in converging as adding A_2 does not lead to convergence to $\mathcal{B}_{UE}$ in an effective manner. In particular, suppose an arc (i,j) has $c_{ij}^d < 0$ and $c_{ij}^D > 0$, then both A_2 and A_3 fail to add (i,j) in the bush. Thereby, the algorithm does not foster convergence as (i,j) is not added, which is not effective in reconstructing bushes due to the reduced cost optimality condition. To add (i,j), one needs the rule A_1 which requires the RMP to be solved to a high level of convergence to avoid the possibility of cycles. This raises another challenge for a bush algorithm: it is hard to justify at which precision level RMPs should be solved to avoid cycles by adding A_1. This study proposes a solution, that is, solving RMPs to ε-optimality, as discussed hereafter.

3.2 ε-optimal bush algorithm (ε-BA)

The concept of ε-optimality (Tardos, 1985) is based on a relaxation of the reduced cost optimality condition. In the TAP context, if a feasible flow vector $\mathbf{x}$ and a dual vector $\boldsymbol{\gamma}$ satisfy the following two conditions, $\mathbf{x}$ is an ε-optimal flow (or ε-optimality) in $\mathcal{B}$:

$$c_{ij}^{\gamma} > \varepsilon \Rightarrow x_{ij} = 0 \qquad \forall (i,j) \in \mathcal{B} \qquad (6a)$$

$$x_{ij} > 0 \Rightarrow -\varepsilon \le c_{ij}^{\gamma} \le \varepsilon \qquad \forall (i,j) \in \mathcal{B} \qquad (6b)$$

The successive approximation algorithm, also labeled ε-optimal bush algorithm (ε-BA), solves ε-optimality as follows.

ε-BA:
Step 0: Initialize a positive scalar λ (both λ and ε are parameters). Initialize a spanning bush $\mathcal{B}$; build an initial feasible solution in $\mathcal{B}$.
Step 1: Rebuild $\mathcal{B}$ by adding a set of arcs that violates the reduced cost optimality condition.

Step 2: Solve ε-optimal flows in $\mathcal{B}$, where ε is a polynomially bounded function of λ.
Step 3: Convergence test. If converged, stop; otherwise improve ε by reducing λ, go to Step 1.

The 0-optimality condition of Eq.(6) equals the reduced cost optimality condition of Eq. (5); and when $\varepsilon \downarrow 0$ (implying $\lambda \downarrow 0$), ε-BA reduces to a bush algorithm. Step 2 differentiates the ε-BA algorithm from a bush algorithm: it solves RMPs to ε-optimality.

In the next section, a cost scaling method is presented to successively refine ε-optimality as part of the ε-BA algorithm.

4. Cost scaling to refine ε-optimality

In ε-BA, Step 2 solves the RMP to ε-optimality in $\mathcal{B}$, which represents the major component of the proposed algorithm. Goldberg and Tarjan (1989, 1990) showed that ε-optimality can be solved for by sending flows along a set of min-mean cycles. Finding a min-mean cycle in a general network requires $O(|N||A|)$ time (Karp, 1978), which is relatively expensive. An alternate method is to apply the cost scaling to augment flows along a set of negative cycles whose mean-value is sufficiently small. In the proposed algorithm, the acyclicity of a bush is leveraged to identify and cancel a set of selected negative cycles whose mean-value is at most a predefined (negative) value $-\lambda$, through a modification of a bush algorithm. Thereby, when there is no negative cycle whose mean value is less than $-\varepsilon$ in the residual network of the bush, ε-optimality is reached in this scaling phase. The algorithm successively reduces λ to refine ε-optimality until λ and ε reduce to zero. At this point, the UE solution is obtained. This cost scaling based ε-optimality method is presented in this section.

The RMP is solved to ε-optimality as follows. Given a spanning bush $\mathcal{B}$ and a positive scalar λ, two shortest path trees are solved with composite costs. $\mathcal{T}_1$ is the shortest path tree in $\mathcal{B}$ with composite cost $c_{ij} + \lambda$; let $\mathbf{u}$ represent the distance label. $\mathcal{T}_2$ is the shortest path tree in the flow-spanned bush $\mathcal{B}_f$ with composite cost $-c_{ij} + \lambda$; the distance label is denoted by $\boldsymbol{\pi}$. Note that both $\mathcal{T}_1$ and $\mathcal{T}_2$ can be solved by one scan of nodes in sequence of the topological order, regardless of possible negative arc costs associated with $-c_{ij} + \lambda$, due to acyclicity of $\mathcal{B}$ and $\mathcal{B}_f$. The nodes in $\mathcal{B}_f$ are scanned in a descending pass. A node k that admits $u_k + \pi_k < 0$ is an eligible node, which identifies a PAS from root r to k along $\mathcal{T}_1$ and $\mathcal{T}_2$. Then, flows are shifted along the PAS such that the travel times along the two segments are equal, or nearly equal. If flow operations have been performed in the descending pass, $\mathcal{T}_1$ and $\mathcal{T}_2$ need to be solved again, and the descending pass is performed again as the arc costs have changed. This process – solving $\mathcal{T}_1$ and $\mathcal{T}_2$ and performing the descending pass – repeats until no node is eligible at this origin, implying ε-optimality of $\mathcal{B}$ is preserved. The approach then proceeds to the next origin, and the process is repeated until no eligible node exists at all origins, implying ε-optimality is preserved at all origins (it is called the ε-optimal phase). Next ε is improved by reducing λ, $\mathcal{B}$ is reconstructed closer to a $\mathcal{B}_{UE}$, and the procedure is repeated again until $\lambda \downarrow 0$ and $\varepsilon \downarrow 0$. The algorithm finally solves an ε-optimal solution restricted in a $\mathcal{B}_{UE}$ at all origins such that the predetermined convergence criterion is met, and stops.

Lemma 1. *Each PAS identified in the algorithm has a mean cost no more than* $-\lambda$.

Proof. Consider a $PAS = \{e_s, e_l\}$; the two segments e_s, e_l are identified on $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. For $(i,j) \in e_s$, there is $c_{ij} + \lambda + u_i - u_j = 0$; for $(i,j) \in e_l$ there is $-c_{ij} + \lambda + \pi_i - \pi_j = 0$, due to the optimality conditions of the shortest path trees. Let k be an eligible node. Then we have $\sum_{(i,j)\in e_s} c_{ij} + |e_s|\lambda = u_k$, and $\sum_{(i,j)\in e_l}(-c_{ij}) + |e_l|\lambda = \pi_k$. The negative cost of the PAS has $\sum_{(i,j)\in e_s} c_{ij} - \sum_{(i,j)\in e_l} c_{ij} + |PAS|\lambda = u_k + \pi_k < 0$, as k is an eligible node. So we have $\frac{\sum_{(i,j)\in e_s} c_{ij} - \sum_{(i,j)\in e_l} c_{ij}}{|PAS|} < -\lambda$. This completes the proof. □

Lemma 2. *When there is no eligible node in $\mathcal{B}_f$, any directed cycle in the residual network of $\mathcal{B}$ has a mean cost at least* $-\lambda - \frac{\delta}{2}$, where $\delta = \max\{u_k + \pi_k \mid k \in \mathcal{B}_f\}$.

Proof. Consider a directed cycle W in the residual network. Partition W into two sets. Let W_1 be the set of arcs (i,j) such that $(i,j) \in A$ and $(j,i) \notin A^-$ in the residual network. Let W_2 be the set of arcs (j,i) such that $(j,i) \in A^-$ and $(i,j) \in A$ in the residual network. For $(i,j) \in W_1$, $c_{ij} + \lambda + u_i - u_j \geq 0$ holds. For $(j,i) \in W_2$, it implies $x_{ij} > 0$, then $(i,j) \in \mathcal{B}_f$ and $-c_{ij} + \lambda + \pi_i - \pi_j \geq 0$ holds. So for arcs in W_1 we apply the first inequality, and for arcs in W_2 we apply the second inequality; we then have $\sum_{(i,j)\in W_1} c_{ij} - \sum_{(j,i)\in W_2} c_{ij} + |W|\lambda \geq \sum_{i\in I(W)}(u_i + \pi_i) - \sum_{i\in O(W)}(u_i + \pi_i) \geq -\sum_{i\in O(W)}(u_i + \pi_i) \geq -\sum_{i\in O(W)} \delta \geq -\frac{1}{2}\delta|W|$ (due to Proposition 1), which leads to $\frac{c_W}{|W|} \geq -\lambda - \frac{1}{2}\delta$. This completes the proof. □

Corollary 1. *When there is no eligible node in $\mathcal{B}_f$, any PAS in $\mathcal{B}$ also has a mean cost at least* $-\lambda - \frac{\delta}{2}$.

Proof. A $PAS = \{e_s, e_l\}$ constructs a directed cycle W in the residual network of $\mathcal{B}$, where $W_1 := e_s$ and $W_2 := \{(j,i) \in A^- | (i,j) \in e_l\}$. The result immediately follows due to Lemma 2. □

Lemma 3. *When there is no eligible node in $\mathcal{B}_f$, the obtained solution is ε-optimal in $\mathcal{B}$, where $\varepsilon = \lambda + \frac{1}{2}\delta$.*

Proof. The relationship between the min-mean cost cycle and ε-optimality is well-established in Theorem 3.3 in Goldberg and Tarjan (1989). Similar proof can also be found in Chapter 10 in Ahuja et al. (1993). To enable the paper to be self-contained, we briefly repeat their proofs.

Denote $\varepsilon = \lambda + \frac{1}{2}\delta$. Consider the composite cost vector $\mathbf{c}' = \mathbf{c} + \varepsilon$. As the mean of the cost of any cycle in the residual network is at least $-\varepsilon$, due to Lemma 2, there is no negative cost directed cycle with respect to $\mathbf{c}'$ in the residual network. The shortest distance label $\mathbf{v}$ with respect to $\mathbf{c}'$ can be computed in the residual network, and $c_{ij}^v = v_i + c_{ij} - v_j \geq -\varepsilon$ holds for each (i,j) in the residual network due to the optimality condition of the shortest path tree.

Given an active arc (i,j) such that $x_{ij} > 0$, it implies both (i,j) and (j,i) are in the residual network. Then, we have $c_{ij}^v \geq -\varepsilon$ and $c_{ji}^v = -c_{ij}^v \geq -\varepsilon$, which leads to $-\varepsilon \leq c_{ij}^v \leq \varepsilon$ and establishes 6(b). The proof for 6(a) is essentially the same as for 6(b). Suppose $c_{ij}^v > \varepsilon$, then x_{ij} cannot be positive as it violates 6(b). The ε-optimality result follows. □

In the above proofs, we justify the ε-optimality but it is associated with a parameter δ. To secure the convergence of the algorithm, that is, secure $\varepsilon \downarrow 0$ when $\lambda \downarrow 0$, we show ε is a polynomially bounded function of λ.

Lemma 4. $\varepsilon \leq |N|\lambda$.

Proof. π_k is the shortest distance label of a node $k \in \mathcal{B}_f$ with respect to the composite cost $-c+\lambda$, so $\pi_k \leq -\sum_{(i,j)\in e_l} c_{ij} + |e_l|\lambda \Rightarrow -\pi_k + 2|e_l|\lambda \geq \sum_{(i,j)\in e_l} c_{ij} + |e_l|\lambda \geq u_k$, because u_k is the shortest distance label with respect to composite cost $c+\lambda$. Then $u_k + \pi_k \leq 2|e_l|\lambda \leq 2(|N|-1)\lambda$ holds for any $k \in \mathcal{B}_f$; that is, $\delta \leq 2(|N|-1)\lambda$. Therefore we have $\varepsilon = \lambda + \frac{\delta}{2} \leq |N|\lambda$ and it completes the proof. □

Lemma 4 indicates that when $\lambda \downarrow 0$, $\varepsilon \downarrow 0$ and ε-BA solves an ε-optimal flow restricted in $\mathcal{B}$. To ensure that a 0-optimal flow in $\mathcal{B}$ is also a 0-optimal flow in G, we need to show that $\mathcal{B}$ converges to $\mathcal{B}_{UE}$ while $\varepsilon \downarrow 0$.

Lemma 5. (Acyclicity condition). *At each ε-optimal phase, a bush $\mathcal{B}$ remains a bush after the set of arcs $A_4 := \{(i,j) \mid c_{ij}^d < -2(|N|-1)\varepsilon, (i,j) \notin \mathcal{B}\}$ is added.*

Proof. We only consider active arcs in $\mathcal{B}$, because we first delete inactive arcs in the bush reconstruction. Any active arc in the network has $c_{ij}^v \leq \varepsilon$. Suppose we add an arc $(k,l) \mid c_{kl}^d < -2(|N|-1)\varepsilon$ that forms a directed cycle W in the network. Consider the subpath p' from l to k in W; then there is $v_k - v_l \leq d_k - d_l + L\varepsilon$, where L represents the number of arcs from l to k along p', as otherwise it violates $\mathbf{v}$ is the shortest distance label with respect to $\mathbf{c}+\varepsilon$. $L \leq |N|-1$. It implies that $c_{kl}^v \leq c_{kl}^d + (|N|-1)\varepsilon$. Then, $c_W = \sum_{(i,j)\in W} c_{ij}^v \leq (|W|-1)\varepsilon + c_{kl}^v \leq c_{kl}^d + 2(|N|-1)\varepsilon < 0$. This is not possible because $c_W \geq 0$ holds for any directed cycle in the network due to the nonnegativity of the arc costs. □

When $\varepsilon \downarrow 0$, $\mathcal{B}$ reconstructed by the rule of Lemma 5 converges to $\mathcal{B}_{UE}$ due to Definition 9. This enables the formal articulation of the ε-optimal bush algorithm:

```
$\varepsilon$-opimtal bush algorithm for the TAP ($\varepsilon$-BA):
1. Initialization:
    Construct an initial feasible solution $\mathbf{x}$ in an initial spanning bush $\mathcal{B}$ for each origin,
    Initialize $\lambda$; set $\varepsilon$-optimality:=false
2. While $\varepsilon$-optimality=false, do (solve $\varepsilon$-optimality at each origin)
    For each origin, do
        Build $\mathcal{T}_1$ in $\mathcal{B}$ wrt $c+\lambda$, and $\mathcal{T}_2$ in $\mathcal{B}_f$ wrt $-c+\lambda$, by using the topological order.
        Do a descending pass of nodes in $\mathcal{B}_f$; for each eligible node $k$, do:
            Identify a PAS along $\mathcal{T}_1$ and $\mathcal{T}_2$; perform one flow operation on the PAS.
    If no eligible nodes in $\mathcal{B}_f$ at all origins, $\varepsilon$-optimality:=true, break the while loop go to
    Step 3; otherwise go to Step 2 repeat the while loop.
3. Convergence check: if it satisfies the stop criterion, stop; otherwise go to Step 4.
4. Reconstruct $\mathcal{B}$ and its topological order at each origin, by adding a set of arcs $A_3 \cup A_4$ (see
   Section 3.1 for $A_3$ and Lemma 5 for $A_4$).
```

$\lambda := \lambda/2$, ε-optimality:=false, go to Step 2.

5. Implementation of ε-BA

5.1 Flow operation

In Lemma 1, we show that a PAS with a mean cost no more than $-\lambda$ could be identified from root r to an eligible node k along $\mathcal{T}_1$ and $\mathcal{T}_2$. In implementation, one can identify a PAS along $\mathcal{T}_1$ and $\mathcal{T}_2$ from the last common node to k, and not necessarily from root r to k. Such a PAS may have a mean cost more than $-\lambda$; however, operating flows on a PAS with smaller size identified from the last common node turns out to achieve a faster convergence consistent with that of a bush algorithm.

With an identified PAS, the method of Bertsekas (1976) and Dial (2006) is used to operate flows along the PAS to achieve the equilibration between the two segments.

$$c_{PAS}(\mathbf{x}+\sigma\mathbf{x}^W) = \sum_{(i,j)\in e_s} c_{ij}(x_{ij}+\sigma) - \sum_{(i,j)\in e_l} c_{ij}(x_{ij}-\sigma) \approx \sum_{(i,j)\in e_s} c_{ij}(x_{ij}) + \sigma \cdot \sum_{(i,j)\in e_s} c'_{ij}(x_{ij}) - \sum_{(i,j)\in e_l} c_{ij}(x_{ij}) + \sigma \cdot \sum_{(i,j)\in e_l} c'_{ij}(x_{ij}) = 0;$$

$$\Rightarrow \qquad \sigma \approx \frac{-c_{PAS}(\mathbf{x})}{\sum_{(i,j)\in e_s} c'_{ij}(x_{ij}) + \sum_{(i,j)\in e_l} c'_{ij}(x_{ij})},$$

where $c'_{ij}(x_{ij})$ is the derivative of $c_{ij}(x_{ij})$ with respect to $\mathbf{x}$.

If the cost function is not differentiable, a binary search or golden section search can be used instead.

5.2 A fast implementation

Preliminary tests of ε-BA on a set of small networks indicate that computationally the most expensive step is that of solving $\mathcal{T}_1$ and $\mathcal{T}_2$. For instance, on the Chicago-Sketch network, it rebuilds $\mathcal{T}_1$ and $\mathcal{T}_2$ 481,815 times, among which in only 94,240 times (nearly 20%) is one or more eligible nodes detected. It indicates that nearly 80% of the computational effort to solve $\mathcal{T}_1$ and $\mathcal{T}_2$ is unnecessary, as it does not involve any flow operations. This trend is consistent for all tested networks in the numerical experiments. This aspect is similar to the findings of TAPAS (Bar-Gera, 2010) – only a few origins are difficult to solve in a general transportation network.

Based on the above observation, a simple heuristic is presented in this section to enhance computational efficiency in the implementation. It has been observed to substantially improve the practical run time performance of ε-BA in all tested networks. It, however, does not imply any theoretical worst-case time bound.

An origin is labeled active if at least one eligible node has been detected in the descending pass; otherwise the origin is inactive. The numerical experiments verify that a number of origins could be inactive. It motivates the following heuristic-based implementation illustrated in Fig. 1. We introduce a number K such that inactive origins are not scanned when zK+1≤iteration≤(z+1)K-1, $\forall z \in \mathbb{Z}$. At the beginning all origins are active, and a descending pass is performed for all active origins. Once an origin is detected as having an empty set of eligible nodes in the descending pass, it becomes inactive, and is not scanned until the Kth iteration. Within the Kth iteration, more origins become inactive and are not scanned in the descending pass as they are more likely to have no eligible node. At every Kth iteration, that is, iteration=zK, $\forall z \in \mathbb{Z}$, all origins are made active again. If all origins become inactive after the descending pass at the Kth iteration, it means there is no eligible node at any origin, and therefore ε-optimality is reached. Otherwise, the process is repeated again. As illustrated in Fig. 1, the implementation does not solve $\mathcal{T}_1$ and $\mathcal{T}_2$ at an inactive origin within the Kth iteration, and thus can substantially reduce the overall frequency of solving for $\mathcal{T}_1$ and $\mathcal{T}_2$, or the number of descending passes.

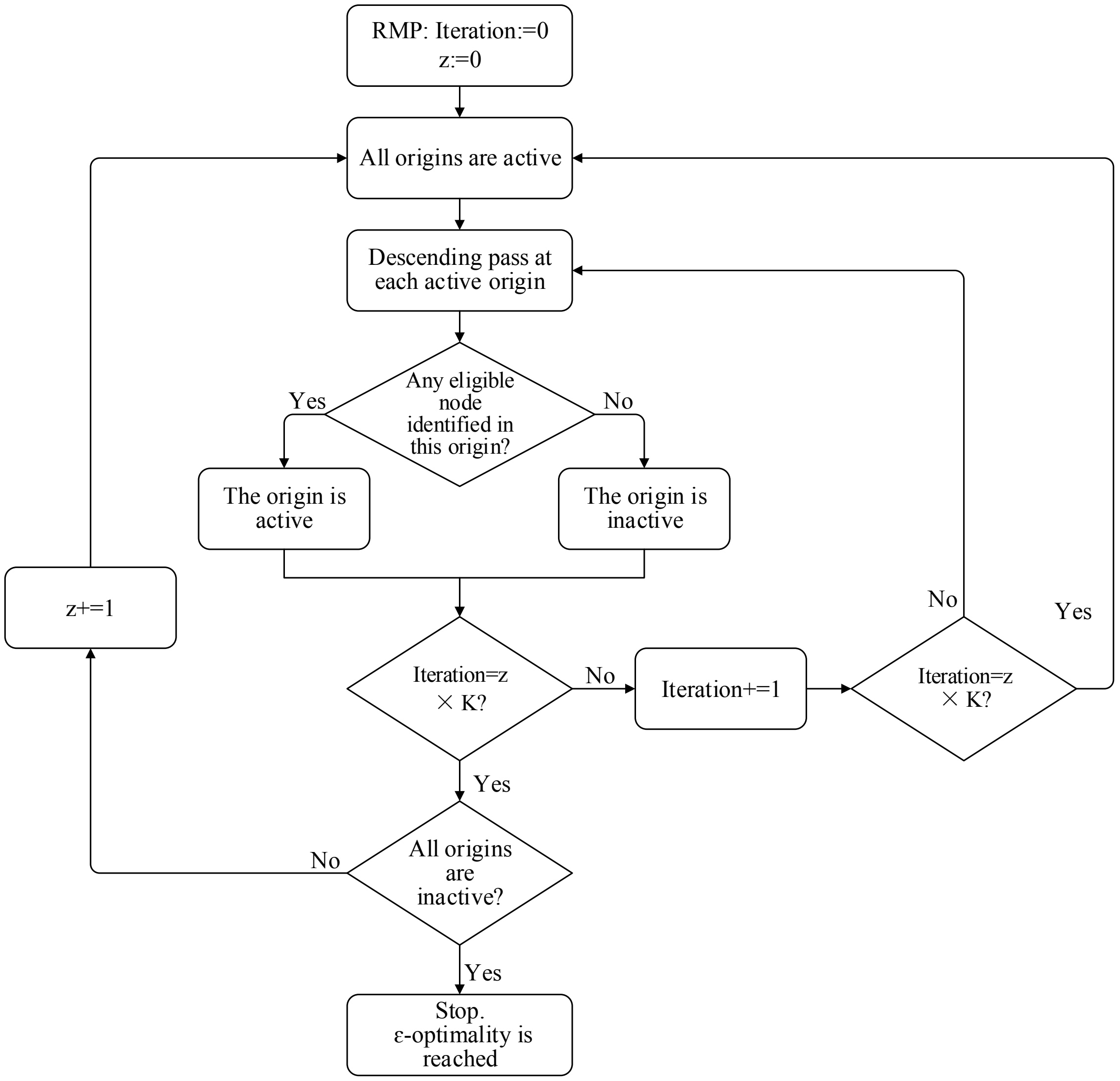


Fig. 1. A fast implementation of RMP in $\boldsymbol{\varepsilon}$-BA.

6. Numerical analysis

Numerical experiments are presented for five test networks accessed from the website

"Transportation Network Test Problems" (Bar-Gera). The characteristics of the test networks are given in Table 1. Three algorithms, identified in Table 2, are implemented in C++ and compared. The computational environment consists of an Intel Xeon E5640 Processor 2.67 GHz with 24GB of RAM. The link cost functions consider the link travel time defined by the BPR-functions, without the consideration of toll factors and travel distance factors.

Table 1

Characteristics of the test networks.

Network	Nodes	Links	Zones	O-D pairs	Trips
Winnipeg	1,052	2,836	135	4,345	64,784
Chicago Sketch	933	2,950	387	93,513	1,260,907
Berlin Center	12,981	28,376	865	49,688	168,222
Chicago Regional	12,982	39,018	1,790	2,297,945	1,360,427
Philadelphia	13,389	40,003	1,525	1,151,166	18,503,872

The convergence of the algorithms is measured using relative gap (RG), defined as follows.

$$\text{Relative Gap}=\frac{\text{total travel time-travel time by assignment on the shortest routes}}{\text{total travel time}}$$

Table 2

TAP algorithms compared in the experiments.

Algorithm	Description
FW	Frank-Wolfe algorithm. The C++ source code is downloaded from the website "Transportation Network Test Problems" (Bar-Gera)
B	Our implementation of Algorithm B by Dial (2006). In our implementation the RMP is always solved 20 times before bush reconstruction.
CS	The proposed cost scaling based successive approximation algorithm (ε-BA). K=10 and λ = 10 are used in the implementation of Fig. 1.

We compare the cost scaling (CS) algorithm with Frank-Wolfe (FW) and Algorithm B. For consistency in comparison, we implement B in a similar way to the fast implementation of CS, as demonstrated in Fig. 2. We say a node is ϵ-UE if the ratio of the difference of longest and shortest distance to the shortest distance is no more than ϵ [3] (that is, travel times on all paths are within ϵ of the shortest path). We say an origin is ϵ-tight for the minimal ϵ for which all nodes are ϵ-UE in an origin-based assignment. The RMP is always solved 20 times before bush reconstruction. When we solve the RMP, we only scan origins which are ϵ-tight and $\epsilon > RG$. We skip the scan of an origin if the origin is ϵ-tight and ϵ is sufficiently small (that is, $\epsilon \leq RG$). Our numerical analyses demonstrate that such an implementation of B not only ensures a fair comparison with the fast implementation of CS, but also helps B converge to a high precision level, say, a RG of 10^{-12}.

Another important factor that significantly impacts the performance of Algorithm B is the threshold value below which flows are assumed to be zero. Due to the floating-point number representation for computing purposes, flows on an arc are assumed to be zero if they are smaller

[3] ϵ-UE here is the same as the ϵ-UE defined in Dial (2006).

than the threshold value. The threshold value should be small enough to preclude possible loss of flow at the desired level of precision. In our implementation, we set the threshold value as 10^{-10}, enabling Algorithm B to converge to a RG of 10^{-12}. If the threshold value is set at 10^{-12}, B converges to at most 10^{-7} on the Chicago Regional network and 10^{-8} on the Philadelphia network. It indicates that the performance of B is very sensitive to the threshold value. By contrast, CS is relatively robust to the threshold value.

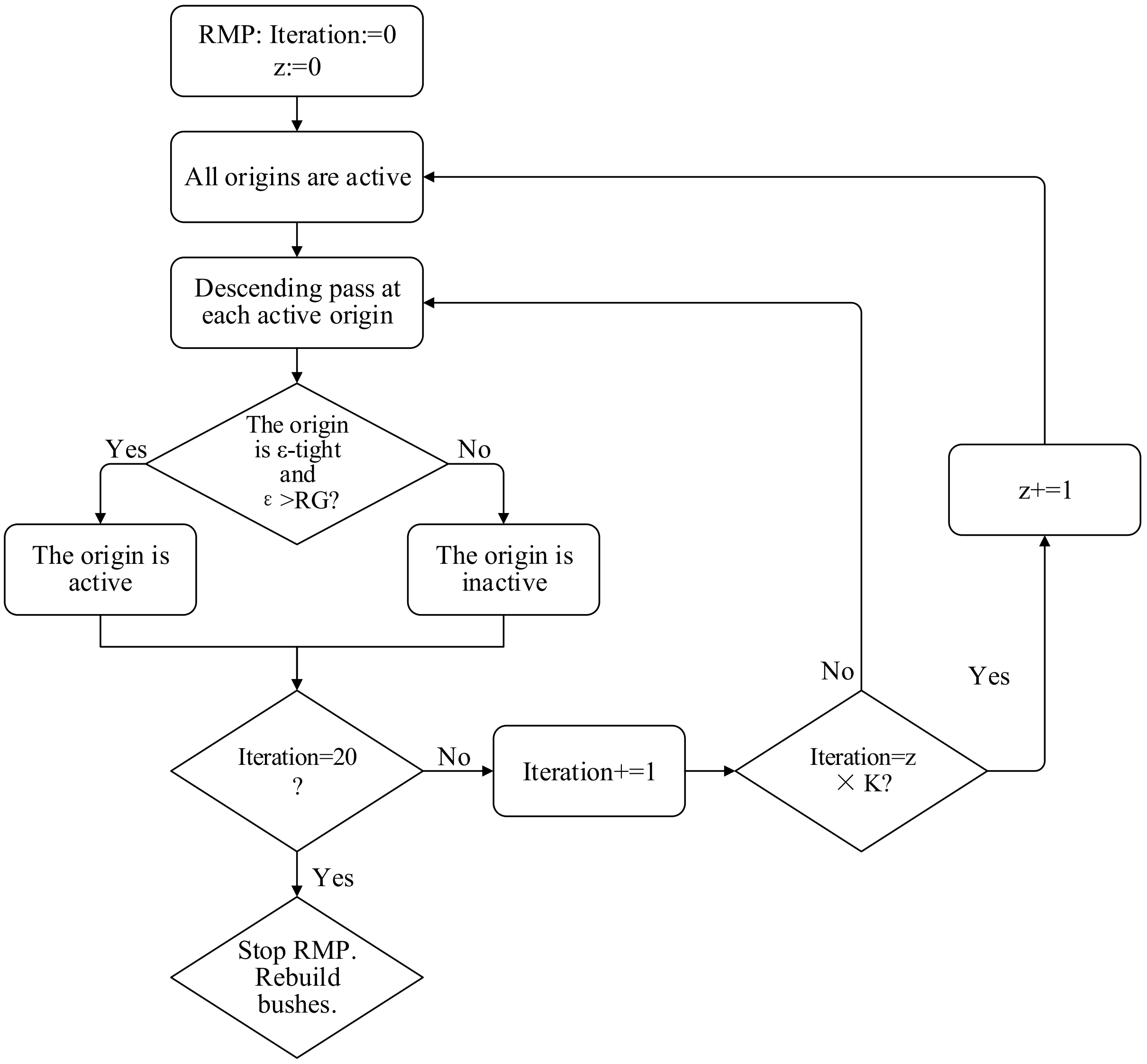


Fig. 2. Implementation of RMP for B.

The choice of the initial value of λ is subject to the following considerations. A higher value of λ implies more scaling phases and usually the identification of a fewer number of cycles when λ is large. This illustrates a tradeoff with implications for the computational effort. A higher value of λ will be meaningful only if there is a significant number of cycles whose mean value is less than $-\lambda$. In our numerical experiments, $\lambda=10$ was found to be an adequate initial value.

The choice of K in Fig. 1 is also subject to a tradeoff. For a larger K, more (less) flow operations are performed on the currently identified active (inactive) origins. As flow operations change the arc costs for all commodities, an inactive origin could become active after flow operations. That is the reason why all origins are made active after every K iterations. Thereby, K should be such that inactive origins are scanned with less frequency while all origins are made active without too long a gap between flow operations. In our numerical experiments, K=10 achieves a satisfactory result.

Table 3 summarizes the computational time for convergence to various relative gaps for the three algorithms. CS can solve the large-scale networks (Chicago Regional and Philadelphia) to RG of 10^{-12} within about 0.5 hours. With the RMP implementation in Fig. 2 and threshold value of 10^{-10}, Algorithm B also converges to a RG of 10^{-12} on the large-scale networks through rule A_3. We note that if either RMP is not implemented as fast implementation, or threshold value is 10^{-12}, then the intermediate bushes do not fully converge to the UE bushes on the large-scale networks. With the fast implementation of RMP, B converges faster than CS for small networks. However, at a high precision level on large-scale networks, CS performs better than B. For small problems, CS may have a disadvantage in that the number of scaling phases can become non-negligible. In such cases, the performance of each scaling phase may be fast, but the overall algorithmic performance after factoring the number of scaling phases can be relatively less competitive.

Table 3[*]

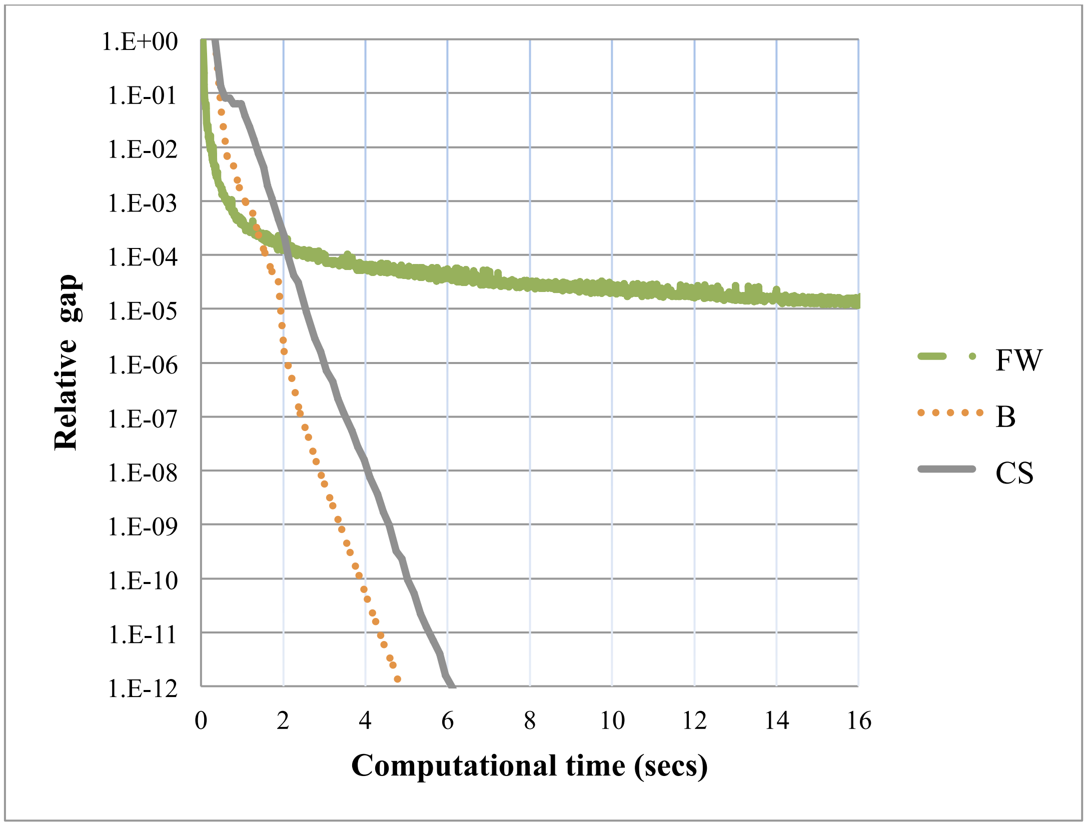
Computational times to convergence for the test networks.

Network	Algorithm	Relative gap					
		10^{-3}	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-12}
Winnipeg	FW	0.7s	2.5s	101.3s			
	B	1.1s	1.7s	2.2s	3.0s	4.0s	4.9s
	CS	1.9	2.1s	3.1s	4.1s	5.0s	6.1s
Chicago Sketch	FW	1.0s	3.4s	268.3s			
	B	2.5s	3.2s	4.3s	5.1s	8.2s	8.6s
	CS	3.6s	4.4s	5.9s	7.4s	9.2s	11.2s
Berlin Center	FW	1.1m	8.2m				
	B	0.5m	0.7m	1.2m	1.2m	2.4m	2.4m
	CS	1.3m	1.5m	2.3m	3.1m	3.8m	4.6m
Chicago Regional	FW	9.5m	50.5m				
	B	4.3m	6.6m	12.7m	22.9m	41.5m	62.3m
	CS	5.6m	7.3m	12.2m	16.2m	20.9m	25.8m
Philadelphia	FW	5.5m	30.2m				
	B	2.8m	4.3m	9.4m	15.8m	19.5m	27.6m
	CS	4.4m	5.2m	8.9m	12.4m	16.2m	19.4m

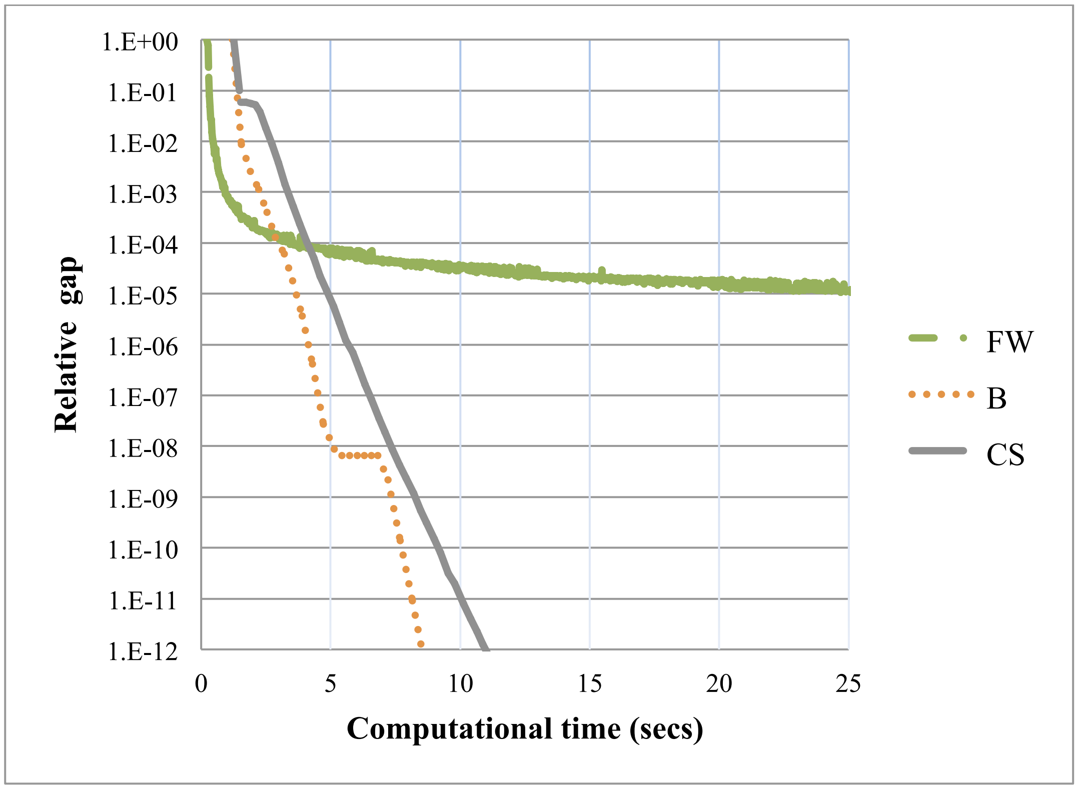
[*] In Table 3, s indicates seconds and m represents minutes.

Fig. 3 further illustrates the evolution of various algorithms in terms of the computational time to various RG levels. It can be observed that the convergence of CS exhibits almost a linear trend, which implies that the effort required to solve ε-optimlaity in each scaling phase has about the same magnitude.

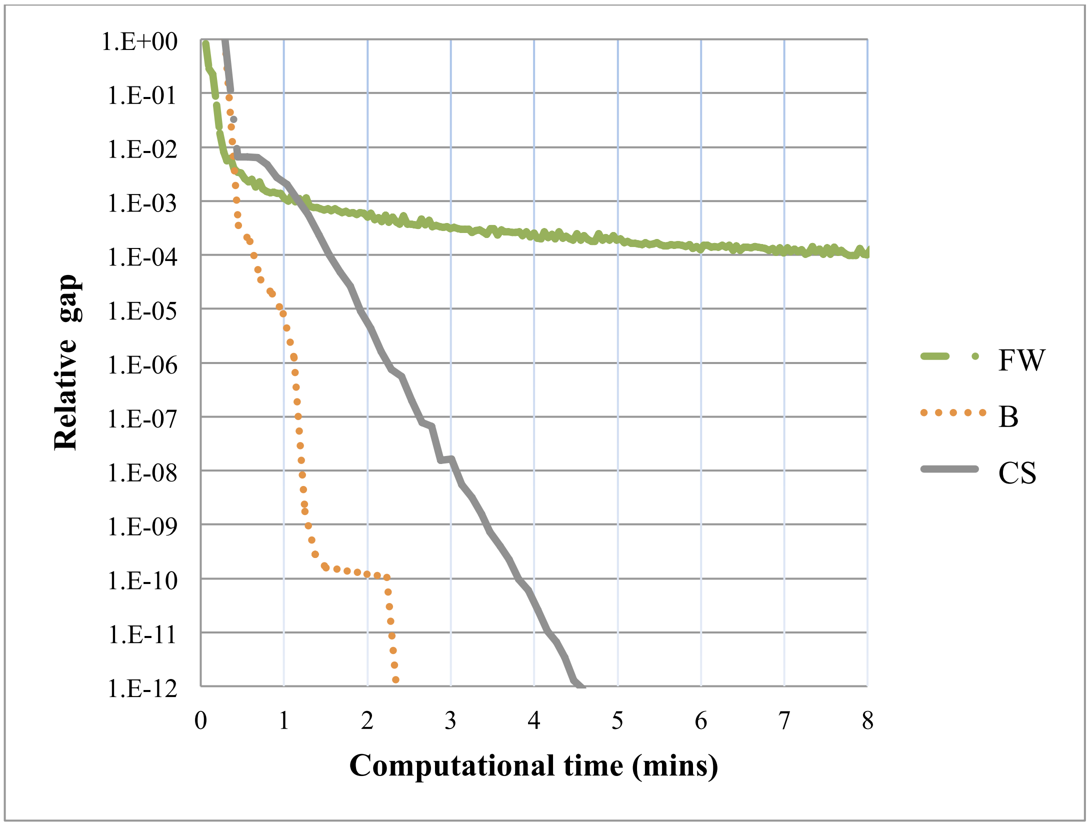
Computational statistics are compared for the Chicago Sketch network in Fig. 4, and the Chicago Regional network in Fig. 5. They illustrate that the frequency of solving for $\mathcal{T}_1$ and $\mathcal{T}_2$ in B is very close to CS. It implies that the fast implementation of B in Fig. 2 can reduce the frequency of solving for $\mathcal{T}_1$ and $\mathcal{T}_2$ significantly.



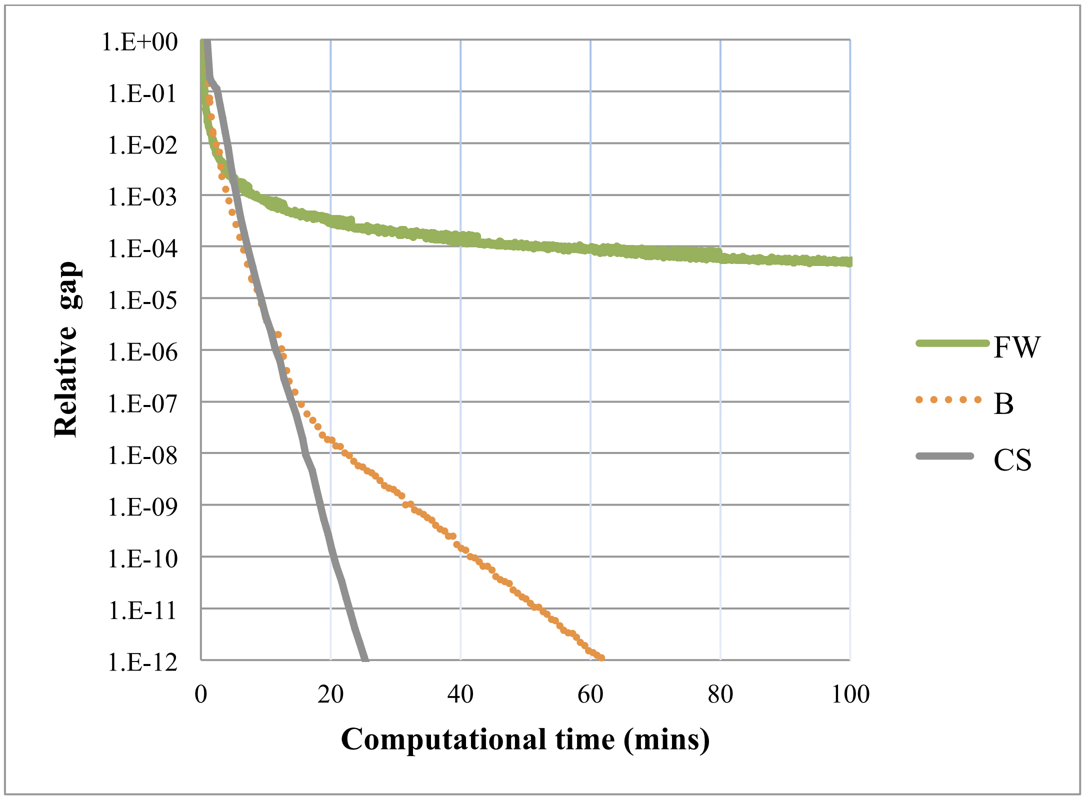
(a) Winnipeg



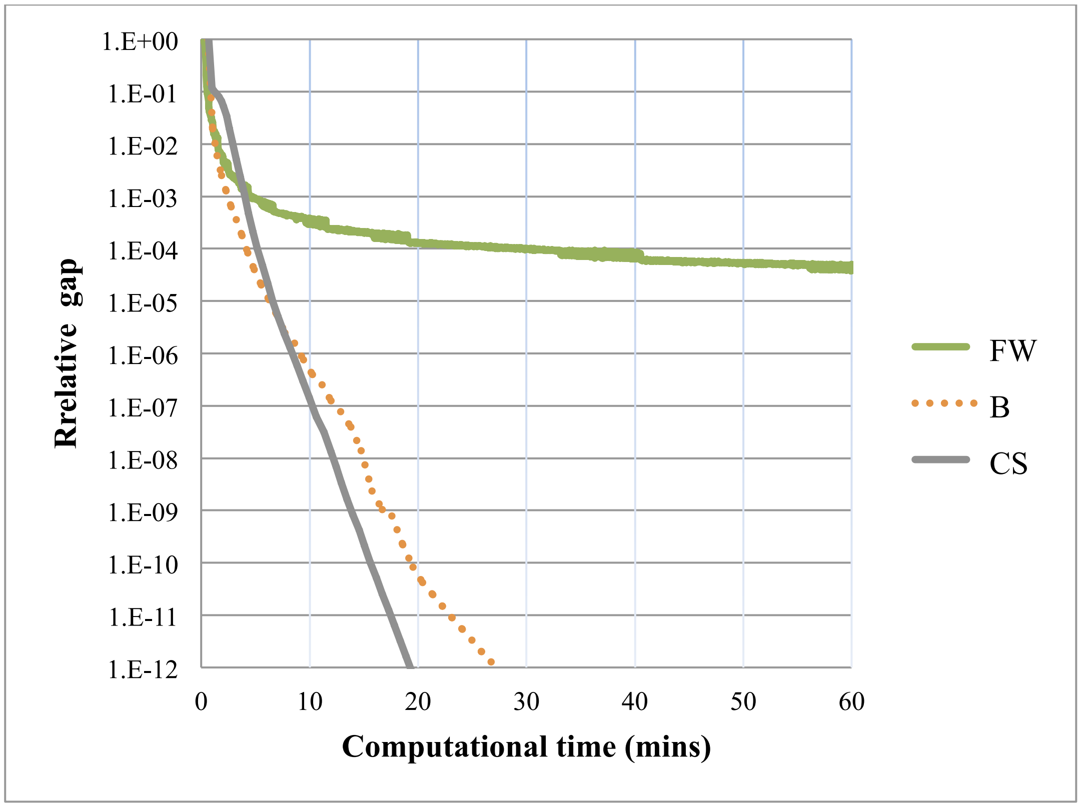
(b) Chicago Sketch



(c) Berlin Center



(d) Chicago Regional

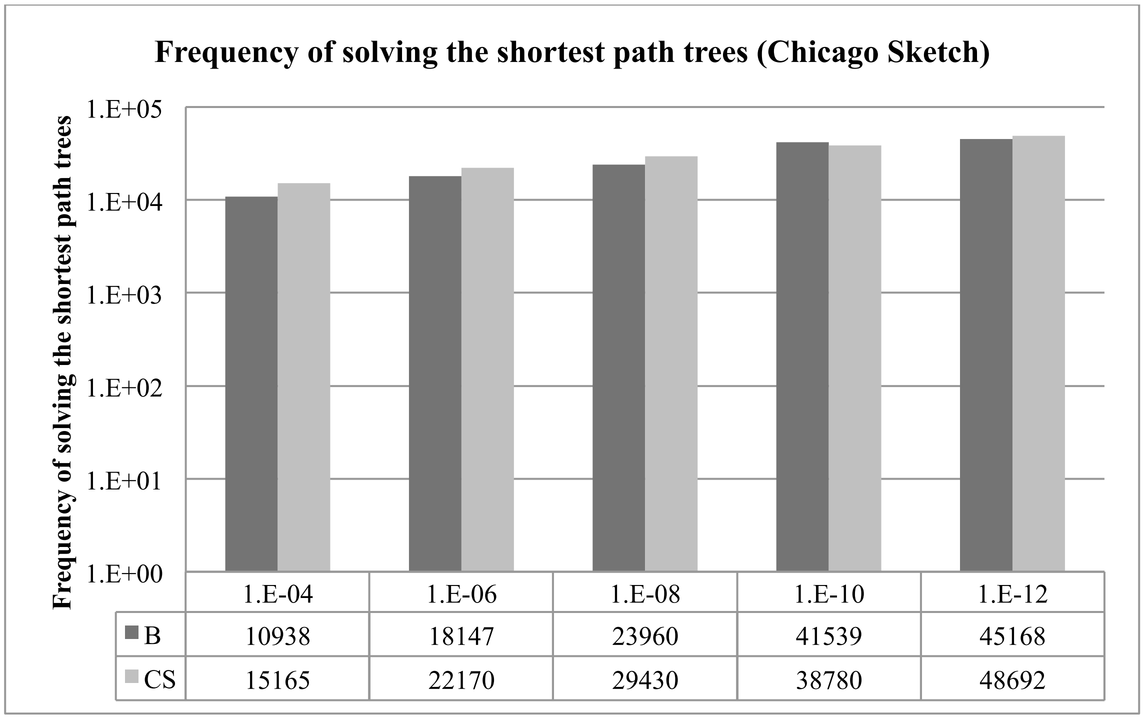


(e) Philadelphia

Fig, 3. Comparison of the algorithms in the test networks.

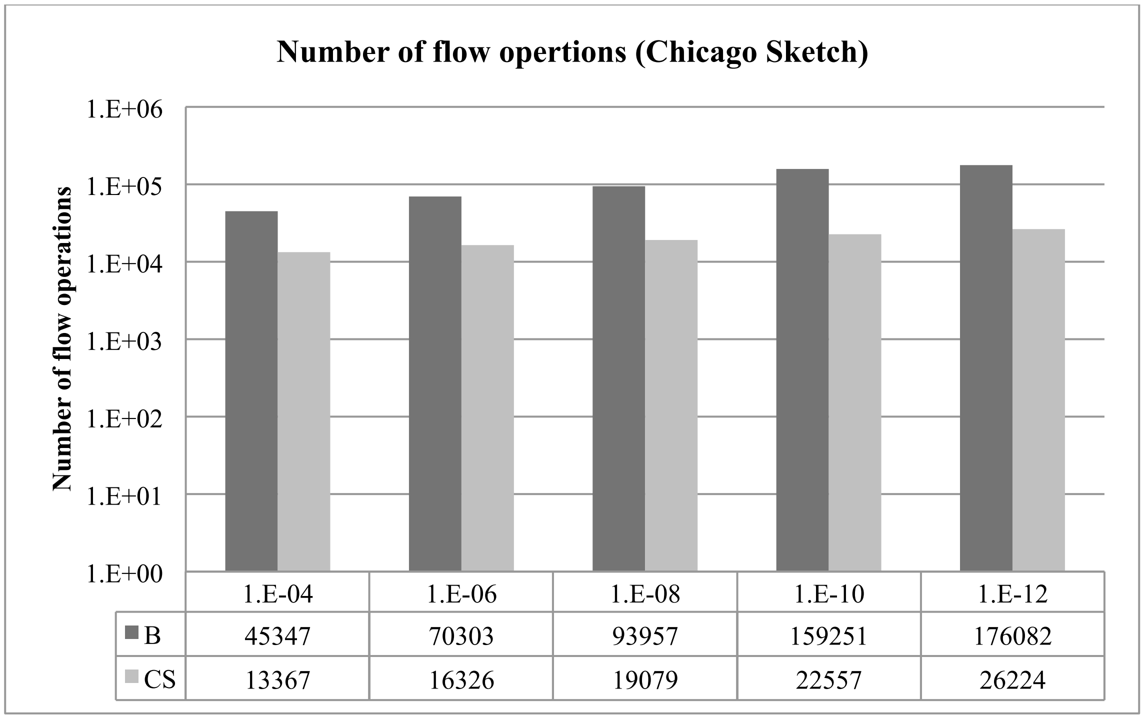
The number of flow operations at different RG levels is compared among B and CS. Fig. 4(b) and Fig. 5(b) show that CS has a lesser number of flow operations. This is because CS augments flows on a set of deliberately selected PAS, and thus can reduce the number of flow operations substantially. On the Chicago Sketch network at RG of 10^{-12}, B has 176,082 flow operations, and CS has 26,224 flow operations. This number is only 15% that of B. The main computational effort of B and CS consists of: (1) the frequency of solving the shortest path trees, and (2) the number of flow operations. Fig. 4(a) and Fig. 4(b) show that the increment in the frequency of solving the shortest path trees is similar for B and CS, but the increment in the number of flow operations at different RG levels for CS is much less than for B.

The effect of the reduced computational effort for CS is much more evident on large-scale networks, which involve large amount of flow operations. On the Chicago Regional network, the number of flow operations by CS is only 8.3% that of B at RG of 10^{-6}. At RG of 10^{-12}, the number of flow operations by CS is 3,337,162 and by B is 140,437,080; hence, CS has 2.3% the number of flow operations of B. These percentages are more evident than under small networks (for example, Chicago Sketch), and again demonstrate that CS can substantially reduce the number of flow operations. This finding suggests an important motivation to design a fast algorithm to solve the TAP in future research; that is, to operate flows with deliberate choices of PAS, or cycles, rather than all qualified PAS or cycles in the network.



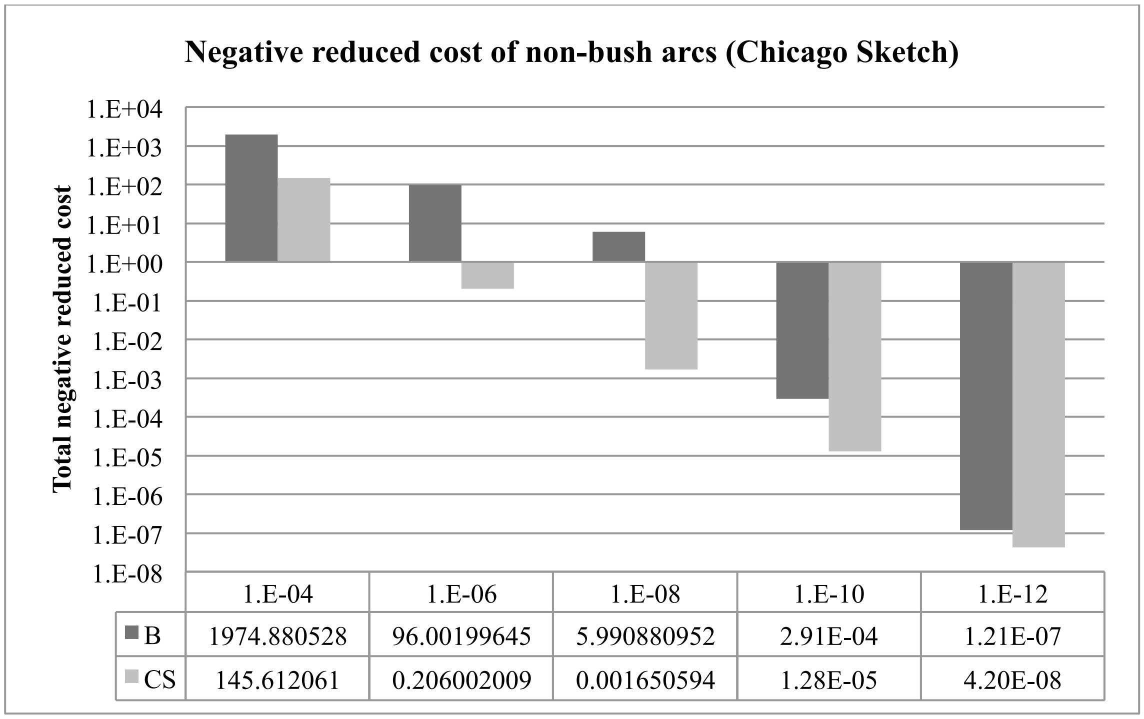
	1.E-04	1.E-06	1.E-08	1.E-10	1.E-12
■ B	10938	18147	23960	41539	45168
■ CS	15165	22170	29430	38780	48692

(a) Frequency of solving shortest path trees vs. relative gap, Chicago Sketch



	1.E-04	1.E-06	1.E-08	1.E-10	1.E-12
■ B	45347	70303	93957	159251	176082
■ CS	13367	16326	19079	22557	26224

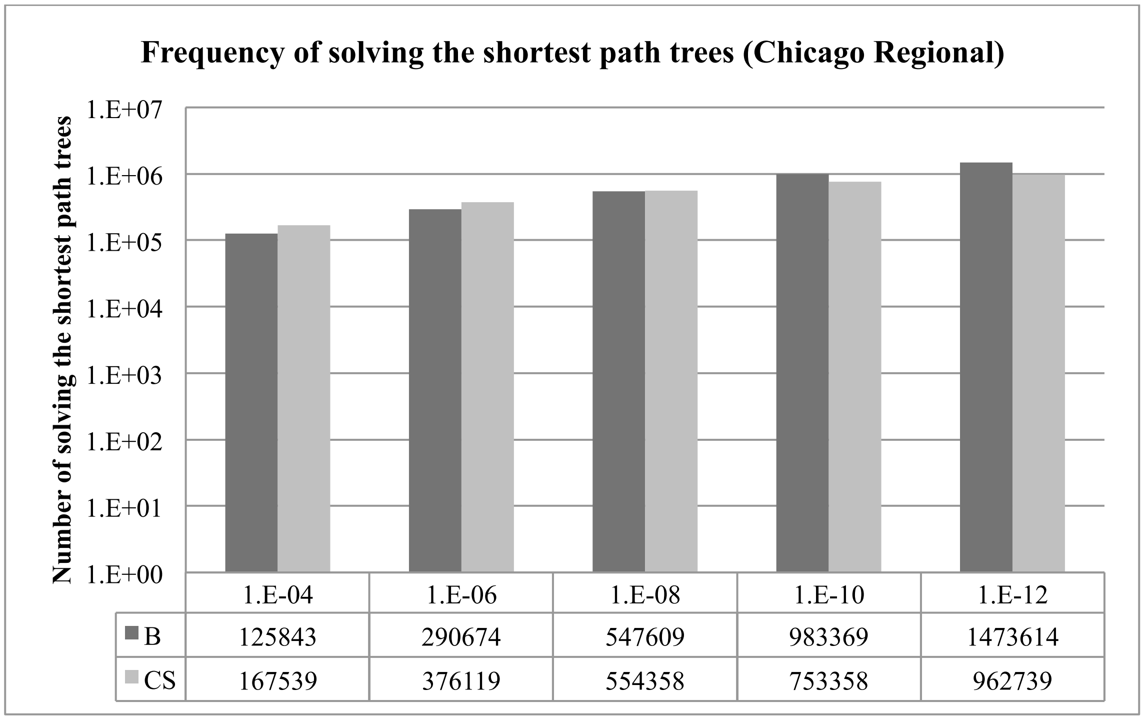
(b) Number of flow operations vs. relative gap, Chicago Sketch



	1.E-04	1.E-06	1.E-08	1.E-10	1.E-12
■ B	1974.880528	96.00199645	5.990880952	2.91E-04	1.21E-07
□ CS	145.612061	0.206002009	0.001650594	1.28E-05	4.20E-08

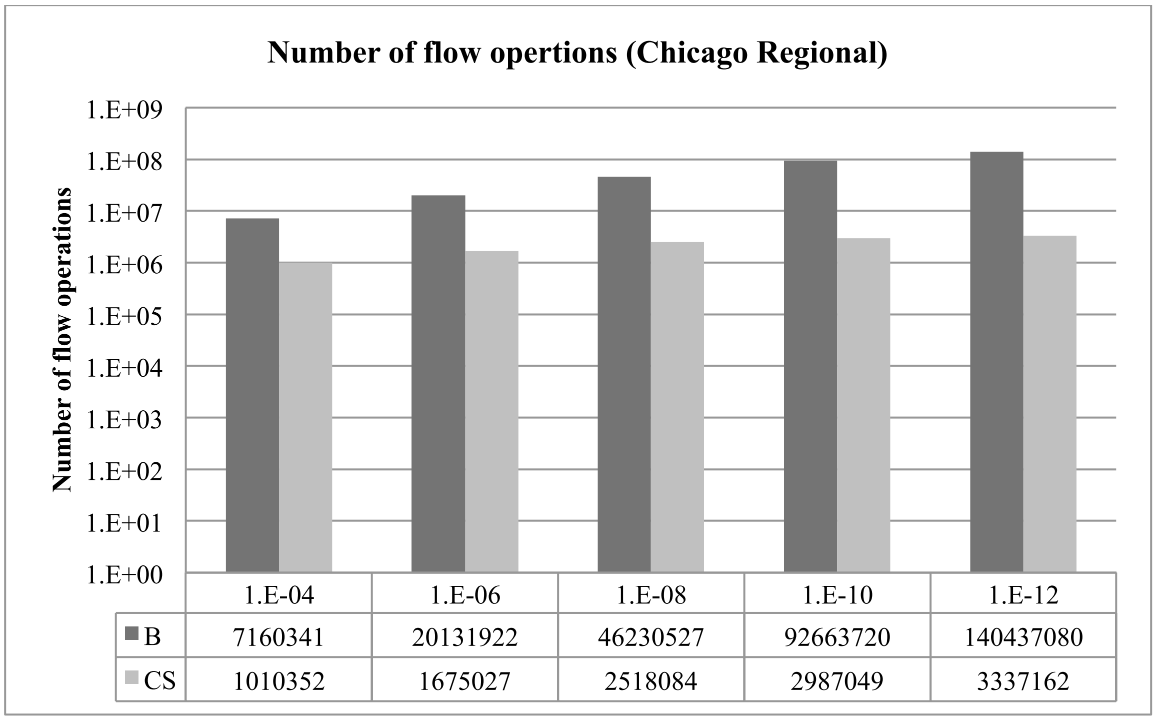
(c) Negative reduced cost of non-bush arcs vs. relative gap, Chicago Sketch

Fig. 4. Computational statistics of algorithms for the Chicago Sketch network.



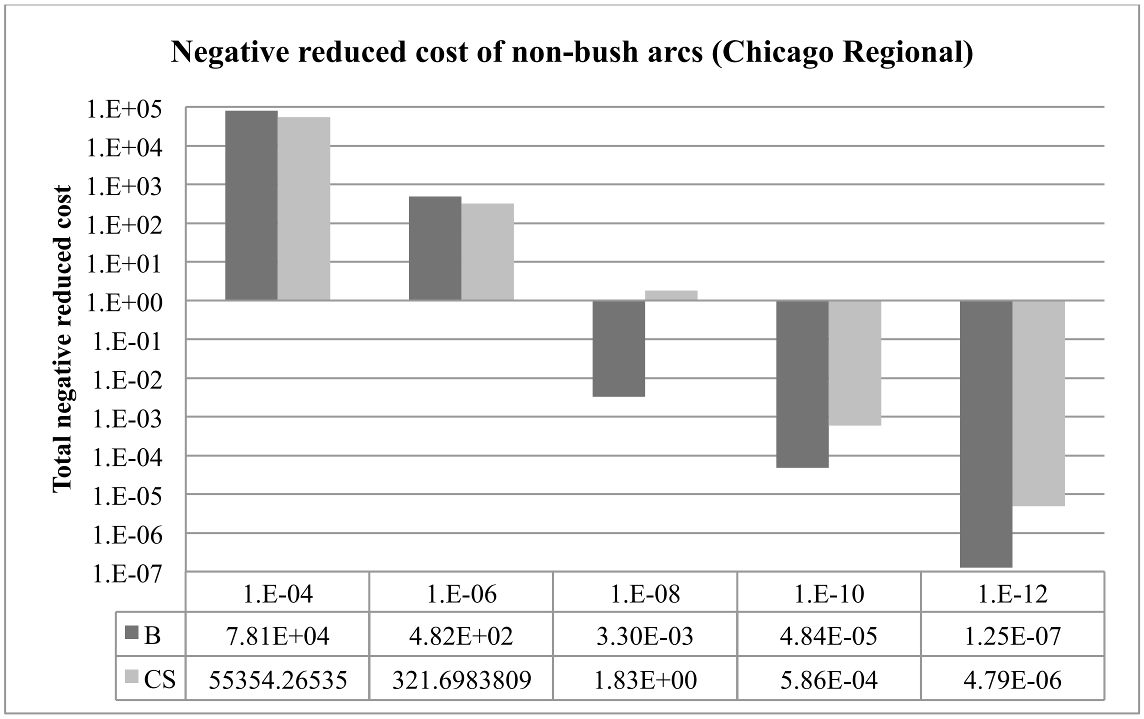
	1.E-04	1.E-06	1.E-08	1.E-10	1.E-12
■ B	125843	290674	547609	983369	1473614
□ CS	167539	376119	554358	753358	962739

(a) Frequency of solving shortest path trees vs. relative gap, Chicago Regional



	1.E-04	1.E-06	1.E-08	1.E-10	1.E-12
■ B	7160341	20131922	46230527	92663720	140437080
■ CS	1010352	1675027	2518084	2987049	3337162

(b) Number of flow operations vs. relative gap, Chicago Regional



	1.E-04	1.E-06	1.E-08	1.E-10	1.E-12
■ B	7.81E+04	4.82E+02	3.30E-03	4.84E-05	1.25E-07
■ CS	55354.26535	321.6983809	1.83E+00	5.86E-04	4.79E-06

(c) Negative reduced cost of non-bush arcs vs. relative gap, Chicago Regional

Fig. 5. Computational statistics of algorithms for the Chicago Regional network.

Fig. 4(c) and Fig. 5(c) plot the total negative reduced costs among non-bush arcs, on the Chicago Sketch and Chicago Regional networks, respectively. The y-axis is plotted on a logarithmic scale. These numbers measure how close the intermediate bushes are to the UE bushes. If the number reduces to zero, it implies that the bushes converge to the UE bushes. In Figs. 4(c) and 5(c), the intermediate bushes in both B and CS converge to the UE bushes ultimately. The figures show that the intermediate bushes in CS are able to converge to the UE bushes by using the rule of Lemma 5; thus CS is able to solve a highly precise solution. For Algorithm B with our fast implementation of RMP and threshold value of 10^{-10}, the intermediate bushes successfully converge to the UE bushes. In our tests, however, if either RMP is not implemented as fast implementation, or threshold value is 10^{-12}, intermediate bushes do not fully converge to the UE bushes. It indicates that the convergence of bushes for Algorithm B is sensitive to several parameters. By contrast, CS is relatively robust.

7. Concluding comments

This paper presents an algorithm to solve the TAP by successive approximation. The proposed algorithm applies a cost scaling method to solve the RMP to ε-optimality, by sending flows along a set of selected PAS whose mean-value is negatively small. Given the acyclicity of a bush, the set of PAS can be solved by the modification of a bush algorithm. The algorithm combines several components, including a min-mean cycle method for the MCF, scaling-like method for successive approximation, and acyclicity of a bush. Through ε-optimality, we show that the intermediate bushes can be reconstructed effectively so as to converge to the UE bushes. Numerical experiments show that the proposed algorithm can solve highly precise solutions on large-scale networks with less computational effort than a bush algorithm, and thus advances the state-of-the-art of the bush-based TAP algorithms.

More generally, the study provides promising insights for the design of faster TAP algorithms. An important finding is that a deliberate choice of cycles to operate flows may reduce the computational effort of the TAP significantly. Min-mean cycles are explored in this study, and are closely related to ε-optimality in successive approximation. The computational effort in terms of number of flow operations can be substantially reduced (to less than one-fifth of a bush algorithm). This finding introduces a new perspective that cycle quality appears critical in the design of faster algorithms for the TAP, which has not been addressed hitherto. Thereby, this study places a premium on the quality of cycles considered to enable a higher level of computational efficiency for the TAP.

References

Ahuja, R.K., Magnanti, T.L., Orlin, J.B., 1993. Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, USA.

Bar-Gera, H. Transportation Network Test Problems, http://www.bgu.ac.il/~bargera/tntp/. http://www.bgu.ac.il/~bargera/tntp/.

Bar-Gera, H., 2002. Origin-based algorithm for the traffic assignment problem. Transportation Science 36(4), 398-417.

Bar-Gera, H., 2010. Traffic assignment by paired alternative segments. Transportation Research Part B: Methodological 44(8-9), 1022-1046.

Beckmann, M.J., McGuire, C.B., Winston, C.B., 1956. Studies in the Economics of Transportation. Yale University Press, Connecticut.

Bertsekas, D.P., 1976. On the Goldstein-Levitin-Polyak gradient projection method. IEEE Transactions on Automatic Control AC-21(2), 174-184.

Bland, R.G., Jensen, D.L., 1992. On the computational behavior of a polynomial-time network flow algorithm. Mathematical Programming 54, 1-39.

Dafermos, S.C., 1968. Traffic assignment and resource allocation in transportation networks. Ph.D. thesis, John Hopkins University, Baltimore, MD.

Dafermos, S.C., 1971. An extended traffic assignment model with applications to two-way traffic. Transportation Science 5(4), 366-389.

Dial, R.B., 2006. A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration. Transportation Research Part B: Methodological 40(10), 917-936.

Florian, M., Constantin, I., Florian, D., 2009. A new look at projected gradient method for equilibrium assignment. Transportation Research Record: Journal of the Transportation Research Board 2090, 10-16.

Frank, M., Wolfe, P., 1956. An algorithm for quadratic programming. Naval Research Logistics Quarterly 3(1-2), 95-110.

Goldberg, A.V., Tarjan, R.E., 1989. Finding minimum-cost circulations by canceling negative cycles. Journal of the Association for Computing Machinery 36(4), 873-886.

Goldberg, A.V., Tarjan, R.E., 1990. Finding minimum-cost circulations by successive approximation. Mathematics of Operations Research 15(3), 430-466.

Jayakrishnan, R., Tsai, W.K., Prashker, J.N., Rajadhyaksha, S., 1994. Faster path-based algorithm for traffic assignment. Transportation Research Record: Journal of the Transportation Research Board 1443, 75-83.

Karp, R.M., 1978. A characterization of the minimum cycle mean in a digraph. Discrete Mathematics 23(3), 309-311.

Larsson, T., Partriksson, M., 1992. Simplicial decomposition with disaggregated representation for the traffic assignment problem. Transoprtation Science 26(1), 4-17.

LeBlanc, L.J., Morlok, E.K., Pierskalla, W.P., 1975. An efficient approach to solving the road network equilibrium traffic assignment problem. Transportation Research 9(5), 309-318.

Nguyen, S., 1974. An algorithm for the traffic assignment problem. Transportation Science 8(3), 203-216.

Nie, Y., 2010. A class of bush-based algorithms for the traffic assignment problem. Transportation Research Part B: Methodological 44(1), 73-89.

Nie, Y., 2012. A note on Bar-Gera's algorithm for the origin-based traffic assignment problem. Transportation Science 46(1), 27-38.

Patriksson, M., 1994. The traffic assignment problem-models and algorithms. V S P International Science Publishers.

Petersen, E.R., 1975. A primal-dual traffic assignment algorithm. Management Science 22(1), 87-95.

Rock, H., 1980. Scaling techniques for minimal cost network flows. C. Hanser, ed. *Discrete Structures and Algorithms*, Munich, 181-191.

Tardos, E., 1985. A strongly polynomial minimum cost circulation algorithm. Combinatorica 5(3), 247-255.