Available online at www.sciencedirect.com
SCIENCE DIRECT®
Transportation Research Part B 40 (2006) 917–936
TRANSPORTATION RESEARCH PART B
www.elsevier.com/locate/trb

A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration

Robert B. Dial *

14820 Oak Vine Drive, Lutz 33559, United States

Abstract

This paper presents a novel user-equilibrium (UE) traffic assignment algorithm, which under conventional assumptions, promises to compute UE arc flows to acceptable precision, regardless of the network's topology, size or congestion:

- The algorithm takes the simple approach of shifting flow from a costliest path to a cheapest path until the costs of all used paths are within a given ϵ of the cheapest.
- Because of being path-based, it avoids tailing.
- In spite of being path-based, it neither stores nor enumerates paths.
- These efficiencies derive from decomposing the problem into a sequence of easy single-origin problems on acyclic sub-networks.

Solutions to this sequence of sub-network flows converge rapidly to a sharp practical estimate of UE arc flows—as is amply demonstrated by tests using the Chicago region's 40,000-arc network model.

1. Introduction

This paper presents a space- and time-efficient path-based solution algorithm for the classical static user-equilibrium (UE) traffic assignment problem (Federal Highway Administration, 1973). In this problem, path cost is an additive function of arc cost, arc cost depends on arc volume, and UE occurs when every trip goes from its origin to destination via a cheapest path (Wardrop, 1952).

1.1. Background

For over a quarter century of transportation planning, the method of choice for computing UE flow aims to minimize a particular nonlinear convex function (Beckmann et al., 1956) subject to linear constraints by

* Tel./fax: +1 813 621 1934.
E-mail address: r.dial@verizon.net

0191-2615/$ - see front matter
doi:10.1016/j.trb.2006.02.008

adapting a quadratic programming algorithm (Frank and Wolfe, 1956), as proposed independently by LeBlanc (1973) and Nguyen (1973).

The good news is that the implementation of this algorithm (FW) is easy to code. Moreover, FW consumes very little memory—an attribute notoriously absent in any prior path-based algorithm. Its simplicity and frugality commend the brilliance of it inventors, whose computer's memory and processor were a million times smaller and slower than our present-day PC's. The bad news is that FW tails badly. No matter the computer, after a few effective iterations, it bogs down into an endless creep, never to reach its objective function's minimum. Its tailing precludes FW's finding a sharp estimate of UE flow in anything except an artificially small or virtually uncongested network.

With urban networks growing larger and increasingly congested, transportation planners should welcome a sharper, faster, and more informative algorithm to complement the large memories and fast processors of today's inexpensive computers. This paper proposes such an algorithm.

1.2. Algorithm B

Dubbed Algorithm B, the algorithm presented here decomposes a UE problem instance on a generally cyclic network into a sequence of origin-restricted UE problems on *acyclic* sub-networks. Using these simpler sub-networks, it efficiently locates and shifts flow from costly paths to the cheaper paths, until all used paths cost within a user-specified ϵ of the cheapest path serving the same origin and destination. Obviating the infamous computer time and storage costs associated with path enumeration, B promises instead several benefits.

1.2.1. Improved precision

Being path-based, Algorithm B computes UE flows with a precision unreachable by Frank–Wolfe algorithms, regardless of the network's size or congestion. Its effectiveness results from exploiting the fact that under traditional traffic assignment modeling assumptions, user-equilibrium arc flow is always at end the simple sum of origin-stratified flow on acyclic sub-networks.

1.2.2. Greater efficiency

Most hard combinatorial issues due to path overabundance disappear when the network is acyclic. Accordingly, B conserves computing time and space by restricting attention to acyclic sub-networks (Dial, 1999b). In doing so, it

- avoids explicitly storing or enumerating paths,
- processes sub-networks containing half the arcs of the entire network,
- builds min- and max-path trees in time linear in this number of arcs, and
- focuses its attention on congested paths.

This results in less computer time and memory usage when compared with that of prior path-based traffic assignment algorithms, e.g., Jayakrishnan et al. (1994) or the 'origin-based' algorithm in Bar-Gera (1999, 2002).

1.2.3. Measure of solution quality

By an approximate UE, we mean a traffic assignment that satisfies a user-specified requirement on the closeness to UE. This specification might take the classical form of an upper bound on the relative gap (see Section 3 below), or of an upper bound on the difference in cost between the cheapest and costliest active (i.e., flow bearing) path pair.

The relative gap γ is the industry standard statistic to assess a traffic assignment's proximity to UE. First proposed and used in the UTPS computer program URoad (UMTA, 1977), is an aggregate statistic equal to one minus the ratio of total travel obtained from the dot product of the arc-flow vector with its associated arc-cost vector divided into origin–destination min-path cost vector times the corresponding demand vector. Accordingly, γ ranges between 0 and 1, with smaller being better. Practitioners consider any traffic assignment with a relative gap less than 0.0001 as excellent; some believe that 0.01 is acceptable. See Boyce et al. (2004) for an extended consideration.

The max–min cost difference bound ϵ is the more intuitive, being directly related to Wardrop's definition of user-equilibrium. For example if given an upper bound of $\epsilon = 0.01$, then an ϵ-UE traffic assignment has all trips using paths that are within 0.01 min of the cheapest path. Given the inevitable uncertainty in arc cost estimation, a traffic assignment with $\epsilon < 0.1$ (6 s) should satisfy any planner. However, until now, this measure has not been accessible.

We would prefer ϵ over γ, not only because the path-cost difference relates directly to the classical definition of UE, but also because real networks are very large while only a tiny fraction of their arcs are congested. This peculiarity causes the relative gap to be over optimistic, especially when congestion is heavy and γ is greater than 0.001. It is likely in this case that many congested paths remain in disequilibrium. The max–min path-cost difference below a specified ϵ would preclude this anomaly. Nonetheless, our test results utilize γ for measuring solution quality.

1.2.4. Pivot-point traffic assignment

A beneficial side effect of Algorithm B is a fast solution to the 'pivot-point' traffic assignment problem (ΔTA). Here a UE solution is known for one trip matrix v and one is required for another matrix differing marginally from the original by a given matrix Δv of positive and/or negative trip differences. The network is invariant; only the trip matrix changes. By easily updating the given solution to serve as a warm start, Algorithm B solves the ΔTA problem in a fraction of the time it would take to compute UE for a matrix $v' = v + \Delta v$ from scratch. Because transportation planning is an exercise in alternatives analysis, this savings accumulates notable benefit. Moreover, it opens the door to sensitivity and uncertainty studies, which could lead to more robust transportation plans.

1.3. Organization of this paper

With this Section 1 of the paper having introduced Algorithm B's features, the three remaining sections overview, verify, and test its design. Section 2 describes and justifies the algorithm informally, temporarily avoiding mathematical formulation in order to maximize accessibility to B's strategy and features. Section 3, on the other hand, is mathematical. It verifies B's convergence and uniqueness properties and provides some implementation details. Section 4 contains examples and test results that compare B's performance with old and new traffic assignment procedures. Section 5 ends the paper with added remarks, comments on related prior work, and acknowledgements.

2. Algorithm B overview

B's effectiveness derives from exploiting the relative ease of computing min and max paths and flow changes on a sequence of origin-rooted flow-feasible acyclic sub-networks, which for short we call bushes:

> A *bush* is a subset of arcs of the original problem's network and comprise an acyclic sub-network rooted at a given origin, together with the arc flows that carry all and only trips from that origin to their specific destinations.

A bush's arcs and its demand-satisficing flow constitute a feasible origin-based acyclic network—a (feasible) bush. An ϵ-UE bush is a bush whose arc flows are ϵ-UE with respect to the paths it contains: We call it ϵ-equilibrated or simply *equilibrated.* A bush whose arc flows are ϵ-UE over all possible paths in the entire network, we call ϵ-optimal, or simply *optimal*.

2.1. Procedure

Driven to achieve a network-wide ϵ-optimal traffic assignment, Algorithm B proceeds as follows:

- *Initialization.* For each origin, create its initial bush and feasible arc flows.
- *General step.* For each origin, transform its current feasible bush into an equilibrated bush:

i. *Build max- and min-path trees:* Find the cheapest path and costliest used path from the origin to each of the other nodes.
ii. *Equilibrate bush:* Shift trips from max- to min-paths to make their path-cost difference minimal.
iii. *Improve bush:* If the bush is not optimal, create a topologically improved feasible bush with feasible arc flows but containing some cheaper path(s).
iv. *Reiterate.* if (iii) changed the bush, go back to (i) with a new bush for the same origin; otherwise, continue on at (i) with next origin's current bush.

- *Termination.* Quit when every origin's bush is optimal.

2.2. Initial feasible bush

To create a bush from scratch is easy: Use free-flow arc costs to build a min-path tree from its origin and compose a sub-network containing only those arcs whose *j*-node is further away (in min-path cost) from the origin than is its *i*-node. Load this bush's min paths with the demand from the origin. Or, alternatively, load the paths with an heuristic that produces a bush that is more equilibrated—a warm start. For example, the initial bushes can come from a prior run of Algorithm B, when the user requires an improved equilibrium.

2.3. Max- and min-path trees

The need for Step (i) is obvious, and the algorithm for finding optimal paths in an acyclic network are well known (Ahuja et al., 1993). Min- and max-path trees build quickly and simultaneously by scanning arcs in the topological order of their *j*-nodes, starting with the root nodes. In this way, each arc in the bush need be visited no more than once per tree pair. Moreover, the cardinality of a bush modeling a piece of road network is approximately half the number of arcs in the network—instead of 4 arcs per node it is about 2 arcs per node; i.e, the min-path tree's arcs plus about one additional arc per node. The last detail to mention is that the max path must only use bush arcs that are *active*—arcs with flow from the bush's origin. Incorporating this constraint poses no problem. In fact it can be used to speed up processing.

2.4. Shifting flow between paths

Setp (ii) merits comment. The min- and max-path algorithms of Step (i) each produce a single *predecessor* arc—the optimal path's arc leading into each node *n*. Flow needs to be shifted only between max- and min-path segments that

i. start at a common node *m*, and
ii. end at a given node *n*,
iii. have no common node between *m* and *n* (Bertsekas, 1976).

In a routine analogous to that used in implementing "simplex on a graph" (Dantzig, 1963), we can find *m* given *n* by backing up along the two trees' predecessor arcs until encountering the first common node. We next employ Newton's method (Bertsekas, 1976) to move quanta of flow from the costlier path to the cheaper path, until either both have equal cost, or one path is cheapest even when carrying all trips the two paths originally shared. In either case, the resulting two path flows satisfy Wardrop's UE criterion in the context restricted to trips populating the current bush.

When the max- and min-path times from *m* to *n* are within ϵ of each other, all paths from *m* to *n* enjoy similar cost propinquity. Because the flow shift neither adds any arc nor alters the *difference* between the inbound and outbound flow at any node, the bush remains feasible. Finally, it is easy to prove (see Section 3) that this simple procedure moves the traffic assignment closer to equilibrium, not only with respect to the current bush but to the entire network.

2.5. Improved bush

Step (iii) warrants explication. At the point when an origin's bush needs improvement. i.e., replacement, it is by definition *equilibrated*—its origin-specific arc flow is ϵ-UE over its own arcs: More precisely, there are no ODs with active path pairs *on the bush* having a cost difference greater than ϵ. For the equilibrated bush to be *optimal*, the min-path node costs *over the entire network* must be within ϵ of their max-path costs *on the bush.* This determination requires merely building a min-path tree over the entire network destinations and comparing its destination-node costs with the bush's corresponding max-path costs.

If the bush is not optimal, Algorithm B improves the bush by swapping out any of its arcs having a (non-bush) mirror with a negative reduced cost: i.e., the old bush's min-path cost to its *i*-node plus the new arc's own cost is less than the old bush's min-path cost to the *j*-node. Perforce, any arc swapped out has no flow. Any arc swapped in also has no flow and gives rise to a cheaper path for one or more ODs, while maintaining the bush's acyclicity.

While bush topology changes, bush arc flow remains nevertheless invariant; therefore, the updated bush topology causes no backtracking with respect to network-wide optimality. The new bush is feasible but not ϵ-UE. Its new arcs, however, introduce cheaper min paths, which will lead to improving its UE estimate. Accordingly, processing returns to Step (i) to again ϵ-equilibrate the same origin, but now with its improved bush.

With the inevitable event that the flow on the bush becomes optimal, the bush cannot be improved. Then, attention shifts to the next origin—which may mean revisiting the first origin—before control returns to Step (i).

2.6. Convergence and uniqueness

The fact that our algorithm converges to the desired UE flow estimate is probably obvious by now. It may not, however, be obvious that the topology of each of the optimal bushes is—as is the optimal *total* arc flow—unique.

2.6.1. Convergence

In theory because Algorithm B is solving a convex program, it must converge to unique UE arc flows. In practice, however, due to numerical issues, it may run forever if given a large highly congested network and an impossibly minuscule ϵ. The solution to a non-pathological problem, however, should arrive in acceptable computer time. There are a finite number of feasible bushes, and with each bush being driven to optimality, better bushes become scarcer with each iteration. Once all bushes are simultaneously optimal, their combined arc flow constitute a unique network-wide ϵ-UE.

Most important, it is easy to prove that the value of Beckmann's minimization function (see Section 3 below) decreases monotonically with each equilibrated bush. In fact, every flow shift contributes to a descending sequence of objective function values representing distinct feasible solutions. This makes thrashing or backtracking impossible. Beckmann's formulation aims to minimize a convex function over a compact set; consequently, Algorithm B descends upon an existent minimum, thus assuring its finding an ϵ-UE.

2.6.2. Uniqueness

Under assumption (i) above regarding the arc cost, Beckmann's formula sums functions that are strictly convex. Accordingly at UE, the total flow on each arc is unique. Path flow and therefore bush flow, however, are not: Multitudinous path flows can add up to the same arc flow. All this is well known.

Less well know is that at UE, the *arc set* comprising an origin's bush is unique. Unique *total* UE arc flow guarantees this lovely topological property. It implies unique arc times, which imply unique min-path topology, which implies unique node potentials, which uniquely determine which arcs populate the bush.[1] The fact that a computer may be incapable to computing exactly these unique UE arc flows and bushes is beside the point. More to point, unique UE bush topology validates Algorithm B's combinatorial aproach: Its sequence of bushes heads for a single landing site.

[1] When both nodes of an arc have equal potential, then their node numbers can be used to break the tie.

3. Algorithm B verification

The prior section aimed to justify Algorithm B with an informal constructive argument. This section provides supporting technical validation. It begins by deriving Beckmann's optimization model (Beckmann et al., 1956), whose objective function is then shown to decrease with each of B's path-pair flow shifts. It concludes that because an ϵ-UE flow for any practical ϵ has a Beckmann function value finitely larger than the minimum, Algorithm B always terminates properly. That is not to say that B's design makes any *operational* use of Beckmann's formulation: His formula's utility here is solely to demonstrate correctness. To this end, we employ the following symbols and their variants:

$\mathscr{G}$	network
$\mathscr{N}, \mathscr{A}, \mathscr{O}$	network's nodes, arcs, and origins (data sets)
$\mathscr{X}$	set of feasible arc flows (traffic assignments)
v_{od}	number of trips going from $o \in \mathscr{O}$ to $d \in \mathscr{N}$
x_{oij}	flow on arc $i \to j$, or $ij \in \mathscr{A}$ for short, that originate at $o \in \mathscr{O}$
x_{ij}	$1^{\mathsf{T}} \cdot x_o =$ total flow on arc ij
$c_{ij}(x_{ij})$	cost of traversing arc ij when its caries flow x_{ij}
$\underline{p}, \overline{p}$	min- and max-path arc sets
x_p, c_p	flow and cost on path p
$\pi_{od}(x)$	min-path cost to go from origin o to destination d wrt arc costs $c(x)$
$\gamma(\mathrm{x})$	$1 - \frac{v \cdot \pi(x)}{x \cdot c(x)} =$ relative gap.

Assumptions. For brevity we make four common assumptions, which facilitate exposition yet do not compromise the algorithm's ability to solve any instance of the classical UE traffic assignment problem:

i. origin–destination trip demand is fixed and non-negative;
i. arc cost is a non-decreasing positive continuous function of its flow;
ii. path cost is the simple sum of its arcs' costs;
iii. arc flows are non-negative real numbers;
iv. every arc has its *mirror*, i.e., for every arc $i \to j$ there is an arc $j \to i$.

Assumption (iv) is the only surprise. Note that while a mirror arc must be present, there is no constraint on its properties—such as it cost function, distance, capacity, etc. For example, to model a one-way street segment, the arc's mirror a very large cost that effectively removes it from use.

3.1. Beckmann's formulation

Lemma 1 (Beckmann). *Given the assumptions (i–iii) above, a user-equilibrium arc-flow vector $x^* \in \mathscr{X}$ minimizes a convex function subject to linear constraints. The objective function sums integrals of arc cost functions*:

$$x^* \in \arg\min f(x) = \sum_{ij \in \mathscr{A}} \int_0^{z_{ij}} c_{ij}(u)\,\mathrm{d}u \tag{1}$$

while $|\mathscr{O}||\mathscr{N}|$ linear equations assure that each trip traverse a path connecting its origin to its destination:

$$\sum_i x_{oid} - \sum_j x_{odj} = \begin{cases} -\sum_d v_{od} & \text{if } d = o \\ v_{od} & \text{otherwise} \end{cases} \quad \text{for } o \in \mathscr{N}. \tag{2}$$

The set of all $x \geqslant 0$ that satisfy (2) define the set of feasible solutions $\mathscr{X}$.

Proof. To verify Beckmann's program, recall that if x^* is UE, then all trips use a cheapest path. Thus, if arc costs were fixed at their optimal value

$$c^* \stackrel{\text{def}}{=} c(x^*) = c(1^{\mathsf{T}} \cdot x_o^*)$$

then x^* must certainly 'on the margin' minimize total travel cost:

$$x \in \mathscr{X} \Rightarrow c^{*\mathsf{T}}x^* \leqslant c^{*\mathsf{T}} \cdot x \Longleftrightarrow c(x^*)^{\mathsf{T}} \cdot (x - x^*) \geqslant 0 \quad \text{for } x \in \mathscr{X}. \tag{3}$$

Consider the above rightmost, variational inequality as a directional derivative, and f's gradient at x^* to be $c(x^*)$. Then any $x \in \mathscr{X}$ that minimizes $f(x)$ satisfies (3) and is UE. This proves the lemma. □

3.2. Convergence

Lemma 2 (Objective function descent). *Assume a network with a component feasible flow $x_o \notin$ UE for some origin node $o \in \mathscr{O}$. Consider two arc-independent paths $\underline{p}$ and $\overline{p}$ that share start and end nodes m and n—not necessarily any trip's original origin or final destination. Let $\overline{p}$ be the more costly of the two paths and be carrying flow $x_{o\overline{p}} > 0$ originating at o, while the flow from o on the cheaper path is $x_{o\underline{p}} \geqslant 0$. Make the flow on the arc set comprising these two paths UE by moving an optimal quantum of flow $\Delta x_o \in (0, x_{o\overline{p}}]$ from $\overline{p}$ to $\underline{p}$. Then this reduces the value of Beckmann's function.*

Proof. Because $x_o \notin$ UE, the path pair $\underline{p}$ and $\overline{p}$ must exist. The objective $f(x)$ readily decomposes into three terms to reveal its separability wrt these paths' arc flows:

$$f(x) = \sum_{ij \in \underline{p}} \int_0^{x_{ij}+\Delta x_o} c_{ij}(u)\,\mathrm{d}u + \sum_{ij \in \overline{p}} \int_0^{x_{ij}-\Delta x_o} c_{ij}(u)\,\mathrm{d}u + \sum_{ij \in \mathscr{A}\setminus(\underline{p}\cup\overline{p})} \int_0^{x_{ij}} c_{ij}(u)\,\mathrm{d}u. \tag{4}$$

Any shift of flow $0 < \Delta x_o \leqslant x_{\overline{p}}$ from $\overline{p}$ to $\underline{p}$ that reduces the sum of the first two terms on the rhs of (4) will reduce $f(x)$. The sum of two terms model a UE sub-problem that assigns a demand of $x_{\underline{p}} + x_{\overline{p}}$ trips going from m to n to a 'network' composed only of arcs in $\underline{p} \cup \overline{p}$. It is not yet optimal, because the path flows $x_{\underline{p}}$ and $x_{\overline{p}}$ are in disequilibrium: $x_{\underline{p}} > 0$ and $c_{\underline{p}} < c_{\overline{p}}$. Achieving this path pair's UE by moving an optimal Δx_o from $\overline{p}$ to $\underline{p}$ retains feasibility, minimizes the target sum, and thus reduces $f(x)$. This proves the lemma. □

Corollary 1 (Flow shift). *Computing the flow shift that puts $\underline{p}$ and $\overline{p}$ in UE entails solving a nonlinear equation with a single positive scalar variable Δx_o. When terms involving Δx_o are differentiable, Newton's method serves solves the problem handily. If derivatives are ill behaved, Golden Section works well instead* (Bazaraa et al., 1993).

Only the arcs in $\underline{p}$ and $\overline{p}$ experience flow change: Shifting Δx_o trips from $\underline{p}$ to $\overline{p}$ impacts arc flows x_{oij} only for arcs $ij \in \underline{p} \cup \overline{p}$. Any change to $f(x)$ thus depends solely on Δx_o. To assure that $x_{oij} - \Delta x \geqslant 0$ for $ij \in \overline{p}$, define $\mu = \min\{x_{oij} | ij \in \overline{p}\}$ and honor the constraint $0 \leqslant \Delta x_o \leqslant \mu$. With x the current total arc-flow vector, and $\underline{c}$ and $\overline{c}$ the cost functions of the two paths (i.e., the sum of their arcs' costs), this two-path UE sub-problem localizes to finding $\Delta x_o \in (0, \mu]$ such that:

i. $0 < \Delta x_o < \mu \Rightarrow \underline{c}(x + \Delta x_o) = \overline{c}(x - \Delta x_o)$,
ii. $\Delta x_o = \mu \Rightarrow \underline{c}(x + \Delta x_o) \leqslant \overline{c}(x - \Delta x_o)$,

which state formally that if possible, path costs are made equal; otherwise the cheapest path's flow is increased by all of the costliest path's flow originating at the current origin. Case ii solves trivially. Case i yields to Newton's method equating the two path costs.

3.2.1. Mechanics

Linearize the path-costs difference:

$$\overline{c}(z - \Delta x_o) - \underline{c}(z + \Delta x_o) \approx \overline{c}(z) - \overline{c}'(z)\Delta x_o - \underline{c}(z) - \underline{c}'(z)\Delta x_o.$$

Set the rhs to zero and solve for Δx_o:

$$\Delta x_o = \frac{\overline{c}(z) - \underline{c}(z)}{\overline{c}'(z) + \underline{c}'(z)}. \tag{5}$$

Substitute the solution to (5) back into $c_{\underline{p}} = \underline{c}(x + \Delta x_o)$ and $c_{\overline{p}} = \overline{c}(x - \Delta x_o)$. Reiterate until the two path costs are equal—being alert that $\underline{p}$ and $\overline{p}$ can swap roles in the process.

Lemma 3 (Acyclic network). *If $x^*_o \in UE$ then all of its active arcs exclusively populate an acyclic bush $\mathscr{G}^*_o = [\mathscr{A}^*_o, \mathscr{N}]$ rooted at an origin node o. That is,*

$$\{ij \in \mathscr{A} | x^*_{oij} > 0\} \subseteq \mathscr{A}^*_o.$$

The flow $x^* \in UE$ is the sum of $|\mathscr{O}|$ such acyclic network flows.

Proof. Since any UE flow satisfies the variational inequality (3), the set of active arcs must contain no cycles. Otherwise, the total cost can be reduced by removing the cycle while maintaining the constraints (2)—by subtracting the smallest arc flow x_{oij} in the cycle from the flow of arcs in the cycle. Thus $x^*_o \in \mathrm{UE} \wedge 0 < x^*_{oij} \Longleftrightarrow ij \in \mathscr{A}^*_o$ for $o \in \mathscr{O}$. □

Lemma 4 (New sub-network implies descent). *Let $\underline{\pi}$ be the min-path potentials of the current bush that is UE restricted to the bush, but not UE over the entire net $\mathscr{G}$. Then the new bush $\mathscr{G}_o = [\mathscr{A}_o, \mathscr{N}_o]$ is defined as all arcs having positive potential difference*:

$$\mathscr{N}_o = \mathscr{N} \quad \text{and} \quad \mathscr{A}_o = \{ij \in \mathscr{A} | \underline{\pi}_i < \underline{\pi}_j\}, \tag{6}$$

leads ultimately to a reduced value of $f(x)$.

Proof. Because min-path potentials always increase along paths from the origin, the new bush spans all nodes and is always acyclic. Moreover, its arc set $\mathscr{A}_o$ includes all prior min paths in the old bush as well as arcs that could improve on those paths. This is, because $c_{ij} > 0$ and arc $\underline{\alpha}_j$'s being in the min-path tree imply that

$$\underline{\pi}_i + c_{\underline{\alpha}_j} = \underline{\pi}_j \Rightarrow \underline{\pi}_i < \underline{\pi}_j \Rightarrow \underline{\alpha}_j \in \mathscr{A}_o.$$

Similarly, for an arc ij to improve some $\underline{\pi}_j$ it must have an arc $ij \in \mathscr{A}$ entering whose reduced cost is negative:

$$c_{ij} - \underline{\pi}_j + \underline{\pi}_i < 0 \Rightarrow \underline{\pi}_i < \underline{\pi}_j - c_{ij} \Rightarrow \underline{\pi}_i < \underline{\pi}_j \Rightarrow ij \in \mathscr{A}_o.$$

Because the new bush yields improved min paths, path-pair cost differences reduce, which in turn leads to a reduction of (1). □

Lemma 5 (Convergence). *Algorithm B computes ϵ-UE in finite time.*

Proof. The feasible set $\mathscr{X}$ is closed and bounded (given), the objective function f is smooth and convex wrt arc flow (Lemma 1), and Algorithm B decreases $f(x)$ each time it equilibrates a pair of min- and max-path flows (Lemmas 2 and 3). Consequently, $f(x)$ value becomes arbitrarily close to the unique minimum $f(x^*)$. Hence an ϵ-UE is reached in a finite number of calculations. □

This ends our proof that Algorithm B terminates with ϵ-UE arc-flow vector. For the Reader requiring more rigorous support than this writer can provide for mathematical claims made here, we suggest Ahuja et al. (1993), Bertsekas (1995), Patriksson (1994) and Rockafellar (1984). Practical examples of B's dogged convergence appear in the following section.

4. Algorithm B test results

This section addresses Algorithm B's performance, as seen in tests with two networks—a tiny toy example and a large real-life case. The toy is a 25-node artificial net; the real-life case is the 40,000-arc Chicago regional network model, The former affords detailed performance analysis whose data would overwhelm us for a network model of realistic scale. The latter yields a candid assessment of B's practicality.

4.1. The toy problem (Figs. 1 and 2)

The toy network appears in Fig. 1. It has four origins/destinations, 25 total nodes, and 40 two-way links—80 arcs with symmetric mirrors. Nodes are numbered, and those in squares—7, 9, 17, and 19—are zone centroids (sources and sinks). The other 21 nodes are transshipment nodes. The matrix at the bottom of the figure shows that all OD pairs have 500 trips. The figure labels each arc in the network with its respective practical capacity and free-flow speed, which enter the BPR formula (Horowitz, 1991) that appears below. Arcs are of two types: Freeways (dark arcs) have a free-flow speed of 55 mph and a practical capacity of 200 vehicles, and others (light arcs) have speed of 20 mph and capacity 300 vehicles. Fig. 2 graphs the arc BPR cost curves for the toy network's two types of arcs. Notice that freeway arcs have less capacity, reflecting the fact that this model tries to account for those arcs' ambient traffic, which is independent of the study period. Another reason is to make the curves intersect and thus when combined with high trip volume, make the problem harder to solve. For comparison purposes, we solve this problem first with the classic Frank–Wolfe procedure, then with Algorithm B.

4.1.1. Frank–Wolfe solution (Figs. 3–5)

FW's processing and solution for this tiny network appear in Figs. 3–5. Even with this toy network, Fig. 3 reveals FW's characteristic quick-start/never-end signature, In fewer than five iterations, FW reduced the relative gap from 0.81 to 0.05, a difference of 0.76. Yet in 95 more iterations, it could reduce it further by only 0.015. FW spent nearly 50 times as long to achieve less than one-tenth of the reduction. If FW were to continue for another 10,000 iterations, it would still not reduce the relative gap below 0.005.

4.1.1.1. FW 'user-equilibrium' (Figs. 4 and 5). Figs. 4 and 5 render active arcs in darker lines. They post arcs times (c^*) and arc flows (x^*), respectively. The value for each directed arc appears on its left. After 100 outer iterations over the entire trip matrix, FW produced an objective function value $f(x^*) = 98{,}615$ along with an admirably small relative gap of 0.0057. The posted arc times and flows look beautiful: As is necessary in this problem instance, flows are perfectly symmetric. Arc times correctly reflect their volume. FW's x^* is obviously a correct UE, right?

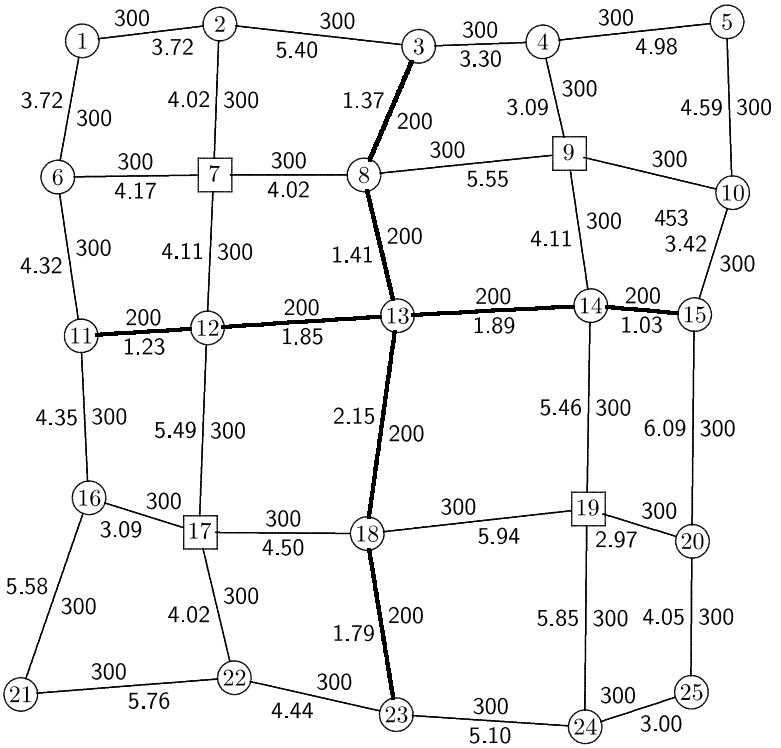
Wrong! There are things amiss in the neighborhood of nodes 17, 18, 22, and 23. The 51 trips on arc $18 \rightarrow 23$ imply that flow goes from 18 to 17 via 23 and 22, and/or from 18 to 19 via 23 and 24, 18 to 17 via 23 and 22. In the former case the path takes 11.05 min, while arc $18 \rightarrow 17$ takes only 8.72 min. In the latter case, the path takes 13.77 min, while arc $18 \rightarrow 19$ takes only 12.01 min. Each path is over 25% longer than its associated min path. FW's flow is far from UE. This difference of a couple of minutes may seem negligible. However, in the context of an 80-arc problem involving an node pair separated by a single arc, it is significant. If errors this large appear in a toy network whose x^* boasts a relative gap of 0.0057, what can we expect in a 40,000-plus network with a relative gap of 0.01? In a word, FW tails early, and its relative gap can be misleading.

4.1.2. Algorithm B user-equilibrium (Figs. 6 and 7)

In the correct UE solution for the toy problem there is no trip from origin node 7 that uses the path $18 \rightarrow 23 \rightarrow 22 \rightarrow 17$ or its mirror. This is seen in Fig. 7, which gives B's answer to the same problem. Its relative gap is 0.00007 vs ODB's 0.0057. B's Beckmann function value of $f(x^*) = 97{,}334$ also differs significantly from FW's $f(x^*) = 97{,}615$. In fact, the lower bound on this function is greater than 97334.40; therefore, B's 97334.4 represents a very sharp UE estimate, whose objective function's value come within 0.01 vehicle minutes of its lowest possible value. An even better metric of solution quality is ϵ, the maximum difference between the costliest used path and the cheapest path. For our toy problem, FW's best estimate of UE $\epsilon > 2.33$ min. In B's solution has $\epsilon < 0.01$ min, as seen in Fig. 6.

Comment: bush transformation (Figs. 8 and 9): These two figure intend to support our earlier claim regarding B's robust ability to find an optimal bush, regardless of its start point:

- *Initial bush for origin 7:* Fig. 8 shows B's initial bush for origin node 7. It labels each node with the min-path cost ($\underline{\pi}$) and the max-path cost ($\overline{\pi}$). Arc labels include the flow in bold face and the cost in normal font. Arrows indicate which arcs are in the bush. Dashed lines indicated bush arcs that do *not* connect



$$c_{ij}(x_{ij}) = c^{o}_{ij}\left(1 + 0.15\left(\frac{x_{ij}}{k_{ij}}\right)^{4}\right)$$

$$v_{od} = \begin{array}{c} \\ 7 \\ 9 \\ 17 \\ 17 \end{array} \begin{array}{c} \begin{array}{cccc} 7 & 9 & 17 & 19 \end{array} \\ \begin{pmatrix} 0 & 500 & 500 & 500 \\ 500 & 0 & 500 & 500 \\ 500 & 500 & 0 & 500 \\ 500 & 500 & 500 & 0 \end{pmatrix} \end{array}$$

Fig. 1. Trip matrix v, arc capacity k and free-flow time c^o.

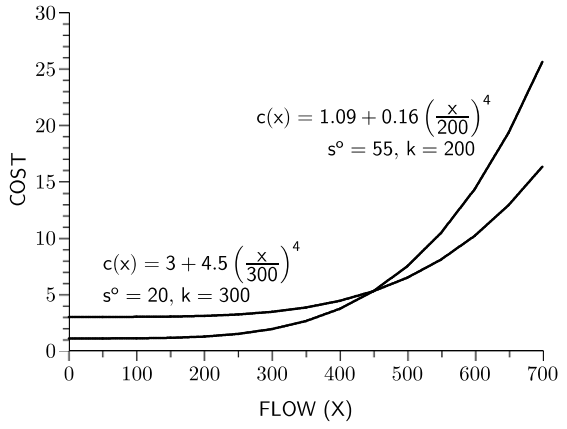


Fig. 2. BPR arc cost vs flow: $C(X|C^o = 60/S^o)$.

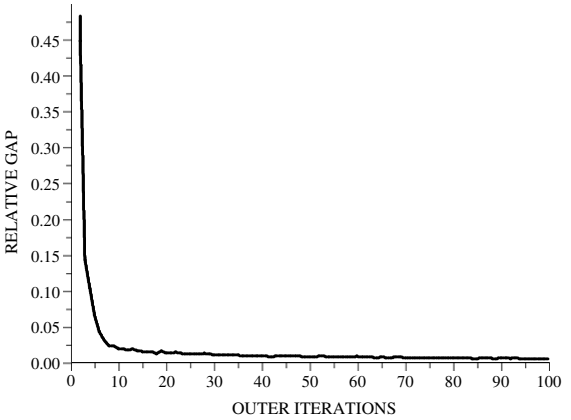


Fig. 3. FW tailing in toy 25-node example.

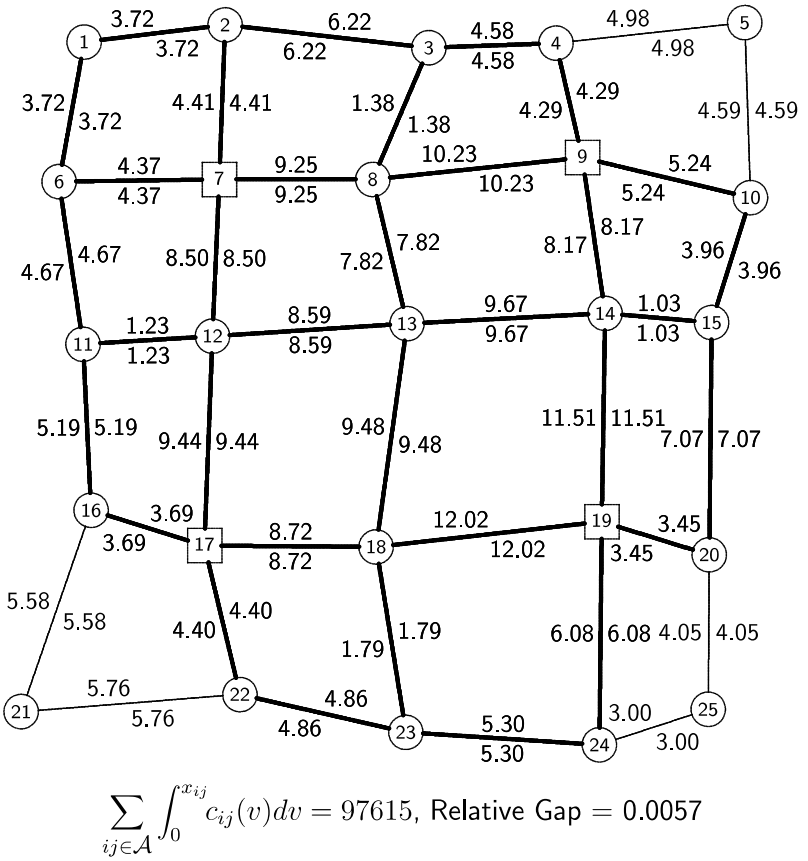


Fig. 4. Frank–Wolfe: "equilibrium" arc costs (flat after 100 iterations).

nodes 7 with any of the three destination nodes, 9, 17 and 14. Consequently, only the solid arcs are useful to B at this stage in finding alternate paths. The largest max–min path difference of $\delta_o = 112.84 - 18.81 = 94.03$ min occurs at node 19.

- *Final bush for origin 7:* Fig. 9 shows the final optimal bush. A visual check verifies that all used paths are minimal and equal to a hundredth of a minute. But we expected this. More interesting is that the final bush

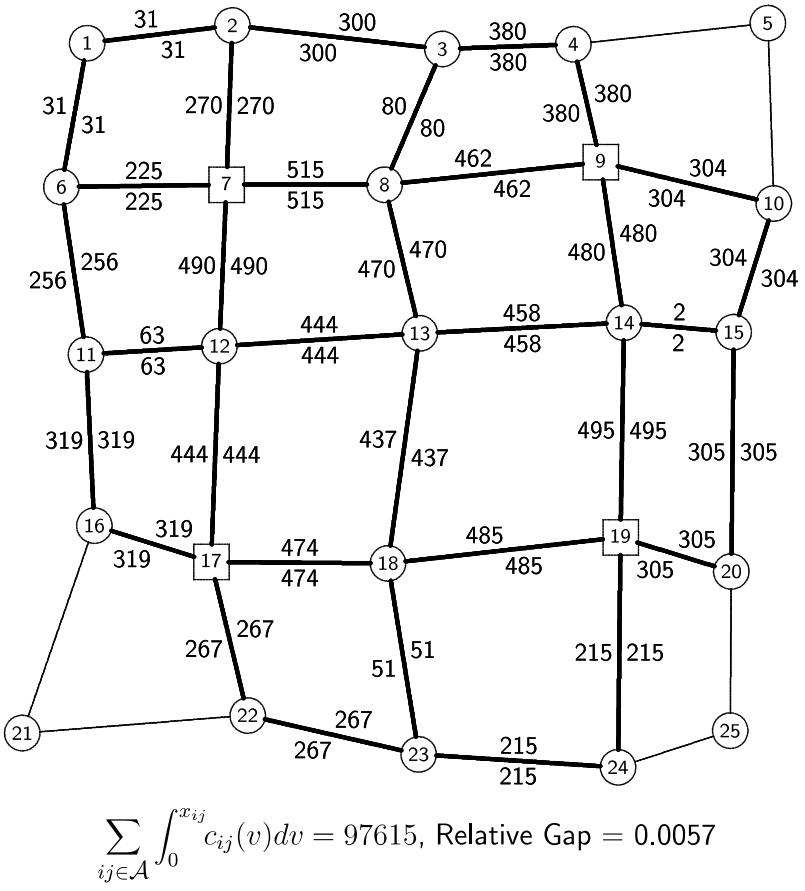


Fig. 5. Frank–Wolfe: "equilibrium" arc flows (after 100 iterations).

differs so greatly from the initial constraining one. Many more arcs are used. Among those absent from the initial bush are: $7 \rightarrow 6$, $4 \rightarrow 9$, $20 \rightarrow 19$, etc.

In our tests, bushes for the other three origins displayed similar transformations, but we lack the space here to include their graphs.

4.2. Results from the regional Chicago network

We now offer several test results using the Chicago regional network model. Constructed by the Chicago Area Transportation Study, this network includes:

- 41,254 arcs, including 2236 (author-added) dummy mirrors,
- 1778 zones, of which 1771 actually produce trips,
- 12,982 nodes, including zone centroids,
- 1,315,989 trips, excluding intrazonals.

4.2.1. Comparing FW with Algorithm B: Fig. 10

Using a 3.0 GHz Pentium 4, we launched FW and B to find a traffic assignment with a relative gap of less than 0.0001. The results appear in Fig. 10. It shows that FW never solved the problem—we shut it off after 5 h. Algorithm B, on the other hand, found the answer in 37 min.

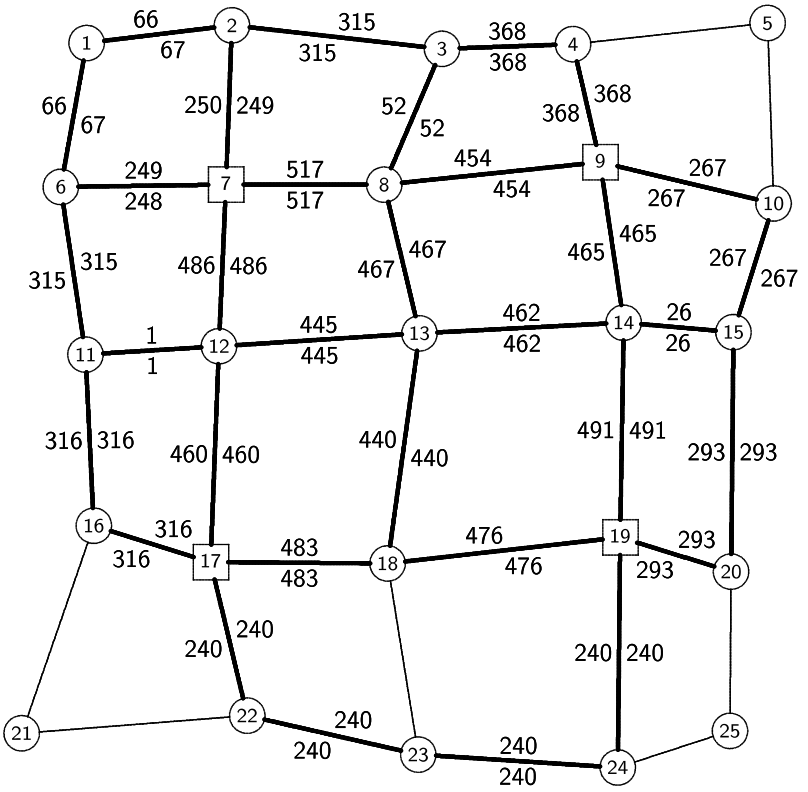


Fig. 6. Algorithm B: equilibrium arc flows.

4.2.2. Stress Testing Algorithm B: Table 1

Having let FW run for 5 h, we thought it fair to run B for a extended period on the Chicago network to see what would happen. Accordingly, we ran it for 284 iterations taking a total of 6.5 h. We took snap shots of B's performance at changes in the magnitude of the relative gap, starting at 6.97×10^{-2} and ending at 9.97×10^{-9}. Because the code was designed to use minimal memory, input–output is relevant: The running time has two components, compute and I/O.

Table 1 shows that in 284 iterations, B build a total of 506,506 min-path trees and 919,798 different bush topologies, from which it computed at least an equal number of min- and max-cost paths. Most notable is that the code performed as hoped: Since trees serve to end iterations, the tree count grew in rough proportion to iterations. Most encouraging are figures showing that 90% of the bushes were built by iteration 4. This adds support to our earlier theoretical claim that Algorithm B is a good bush pruner. (In fact, a recently improved implementations has sped up bush pruning by an order of magnitude.) A practical observation is the negligible change in Beckmann's function after the relative gap reaches 10^{-6}: From a planner's perspective, the additional 200-odd iterations wasted computer and clock time.

4.2.3. Iterations vs relative gap: Fig. 11

As expected, Table 1 reveals B's non-linear relationship between its iteration count and final relative gap. While not nearly as grotesque as FW's tail, B's suggests nevertheless that adding a decimal to the relative gap means doubling its iteration count. Given the reasonable 37 min B takes achieve an enviable gap of 0.00009, this observation would be of marginal interest, were it not such a good fit. To see if a generalization was

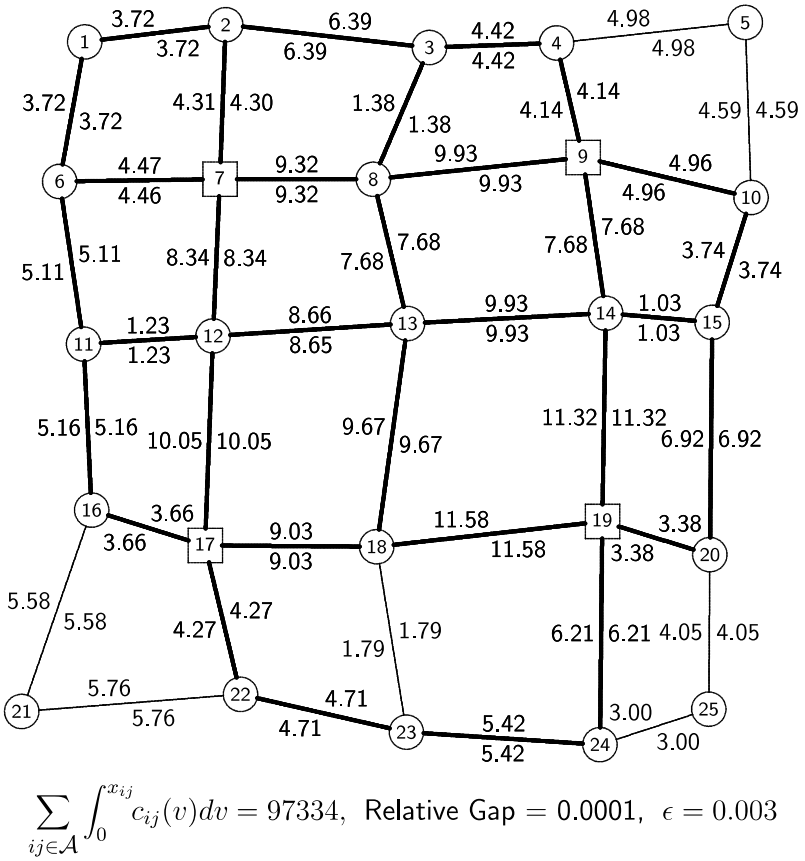


Fig. 7. Algorithm B: equilibrium arc times.

surfacing, we ran B on the toy network again, this time seeking the minuscule relative gap of 10^{-10}. We took snapshots and plotted B's iteration count vs the toy's relative gaps alongside the same data for Chicago. The graphs matched. Both data sets suggest that to divide the gap by 10 means to multiply B's iterations by two. Fig. 11 shows the results. The y-axis is logged to base 2 and the x-axis to base 10—in negative orientation. Given the disparity of the networks, it is an interesting result.

4.3. Comparing Bar-Gera's OBA with Algorithm B: Fig. 12

Having earlier addressed FW, we now compare Algorithm B with a more recently proposed algorithm by Bar-Gera (2002), which he calls origin-based traffic assignment (OBA). The Open Channel Foundation website http://www.openchannelfoundation.org/projects/Origin-Based_Assignment/ provides OBA's executable computer code as well as the Chicago regional network model and trip matrix. Engineers at Caliper Corporation downloaded that code and data, and tested Bar-Gera's own code implementing his algorithm and running on his test data to assess OBA's performance in achieving various relative gaps. They reported their results at the 2005 Planning Applications conference of the Transportation Research Board (Brandon et al., 2005). In addition, Caliper tested this writer's executable code for Algorithm B using the same data and machine they used to test OBA—a 2.0 GHz Pentium M, which is faster and has more memory than my own PC that ran the tests reported on up to now.

Graphing the results of Caliper's two tests using the Chicago regional network model, Fig. 12 compares the performance of Algorithm B and OBA. The figure shows running times to achieve three relative gap thresh-

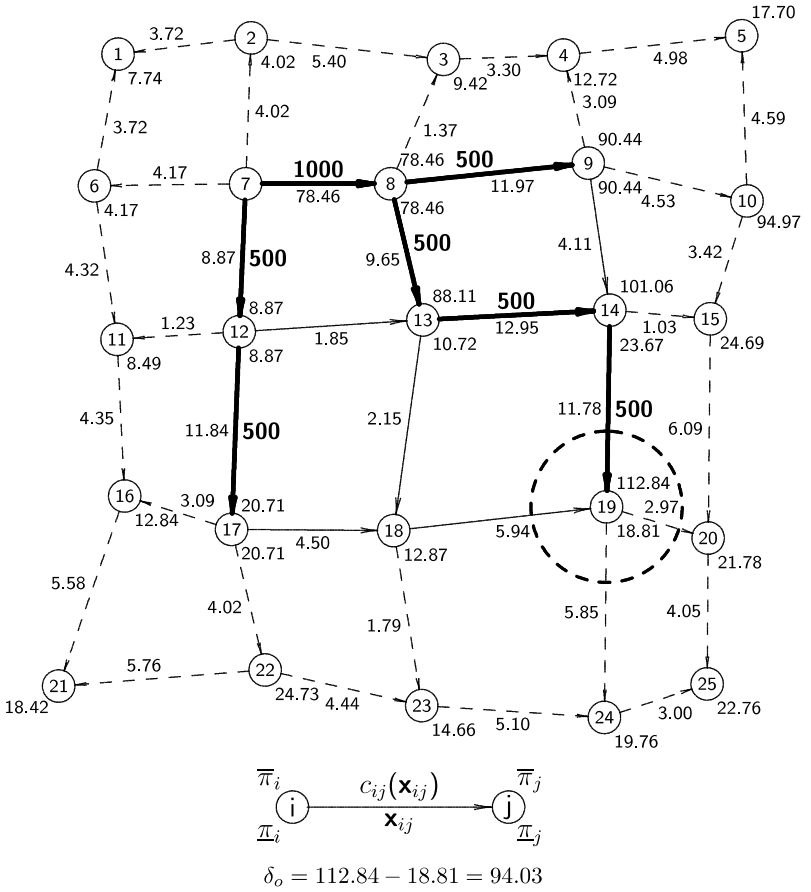


Fig. 8. Origin = 7 bush: initial topology and flow.

olds: 0.01, 0.001 and 0.0001. Algorithm B soundly beats OBA at every relative gap. And the tighter the gap, the sounder the beating:

- to reach a gap of 0.01 Bar-Gera's OBA takes 16 min, while B takes 9 min;
- to reach 0.001 this difference increases, with OBA needing 66 min and B needing only 15;
- to reach 0.0001 OBA's performance takes 175 min vs B's 29 min.

Remarkably, for a gap below 0.0001, Algorithm B solves the 40,000-arc Chicago network nearly six times faster than OBA. Further evidence of B's superior efficiency came when Caliper applied OBA to our 80-arc toy problem: OBA failed—after 1000 iterations—to match the UE precision that B obtained in 50-iterations (Rabinowicz, 2005).

4.4. Pivot-point traffic assignment: Table 2 and Fig. 13

We conclude this section with statistics supporting our previous remark: Given a UE solution for one trip matrix v together with new matrix $v + \Delta v$ differing marginally from the old, B can use the old's solution find a solution for the new matrix in notably less time than starting from scratch. Recall that the network is assumed invariant; only the trip matrix changes.[2] We synthesized seven $v + \Delta v$ matrices by factoring the Chicago trip

[2] Minor change in the network's arc costs is also supported; however, we do not discuss or test this feature here.

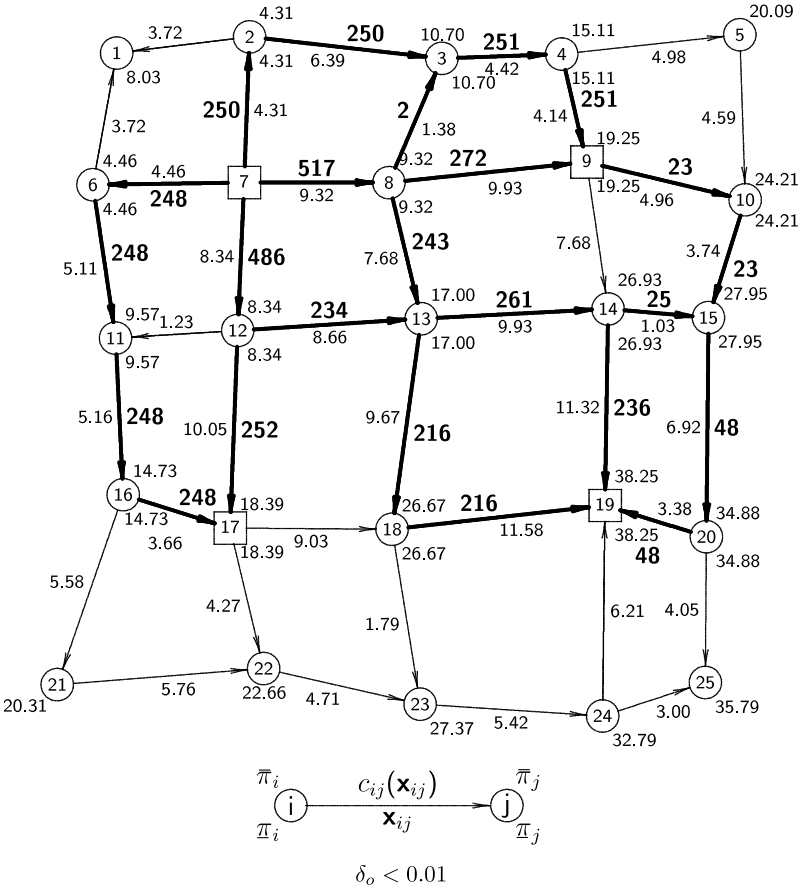


Fig. 9. Origin = 7 bush: final topology and flow.

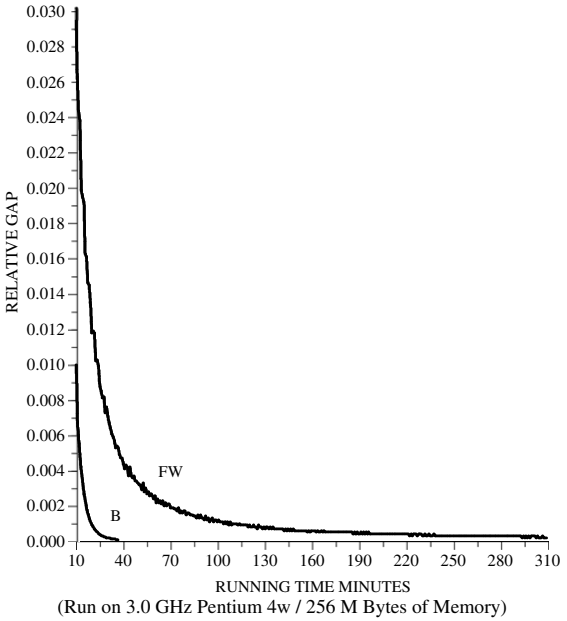
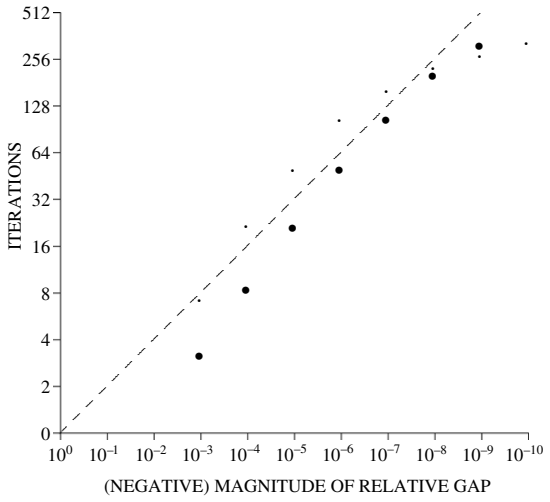


Fig. 10. FW in Chicago 12,982-node example.

Table 1
Stress test iteration summary

Iteration	Elapsed minutes			Total		Citeria	
	Total	Compute	I/O	Trees	Bushes	Relative Gap	Beckmann
1	2.56	2.49	0.07	3,542	909,115	0.0697	26891552.28
4	11.51	10.48	1.03	8,855	913,228	0.00668	25931132.66
9	21.34	18.73	2.60	17,710	916,828	0.000914	25855798.49
20	39.34	32.93	6.41	38,962	918,586	0.0000901	25847329.33
47	77.52	62.66	14.86	86,779	919,410	0.00000924	25846943.36
99	149.68	118.38	31.29	178,871	919,719	0.000000980	25846916.89
190	274.37	214.33	60.04	340,032	919,792	0.0000000986	25846915.62
284	398.22	308.57	89.64	506,506	919,798	0.00000000997	25846915.59

Run on 3.0 GHz Pentium 4 with 256MB of RAM, Chicago regional network: 1778 zones, 12,979 nodes, 39,017 arcs.



· represents 25-node toy network data.
• represents 12,982-node Chicago network data.

Fig. 11. Iterations vs magnitude of relative gap.

matrix, producing seven trip matrices that decreased the original by 5%, 10%, and 20% and increased it by 5%, 10%, 15% and 20%. All tests aimed to achieve a new assignment with the same relative gap threshold as the original 0.0001.

Table 2 and Fig. 13 graph the resulting running times, both of starting from scratch (cold start) and from arc flows of UE assignment of the original matrix (warm start). With the same warm start serving each of the seven tests, speedups ranged from 1.89 to 3.93. As expected, the speedup improved with smaller $|\Delta v|$. (The case where the factor is 1.00 is the null change, tested as a sanity check. Algorithm B required its 3-min initialization to realize it was already at UE.) As do cold starts, warm starts take longer to solve when total volume increases to exacerbate congestion.

Note that the time to run all seven new traffic assignments from scratch totaled 4.83 h; whereas the warm start, reduced this time to 1.83 h. This resulted from the fact that in contrast with the FW's output—an arc-flow vector summed over all origins—B's output is a bush for each origin. Besides their enabling pivot-point traffic assignment with its attendant time savings, these origin-stratified arc flows can potentially provide better information for highway design, maintenance priorities, environmental impacts, toll roads, etc.

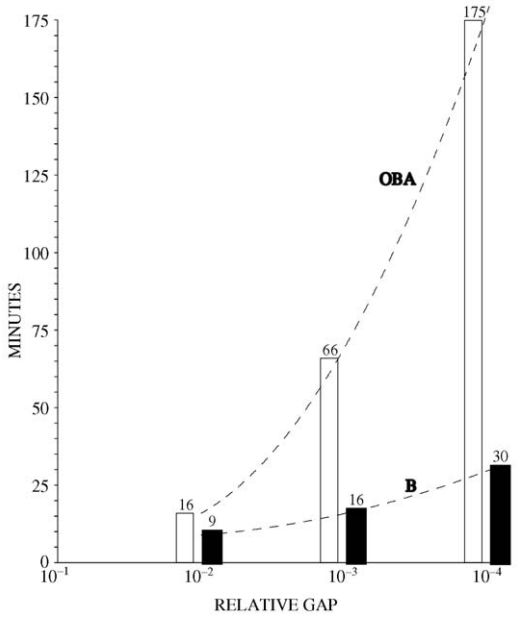


Fig. 12. Execution time: OBA (white) vs Algorithm B (black).

Table 2
Time to reach $\gamma < 0.0001$: cold vs warm start

Factor	Cold start			Warm start		
	Iterations	Relative gap	Minutes	Iterations	Relative gap	Minutes
0.80	13	0.000085	23.78	6	0.000090	12.54
0.90	16	0.000091	29.19	6	0.000082	11.73
0.95	18	0.000086	32.68	4	0.000095	8.35
1.00	25	0.000091	37.45	1	0.000078	3.26
1.05	23	0.000092	41.64	6	0.000088	11.58
1.10	25	0.000091	45.24	9	0.000084	16.81
1.15	28	0.000091	50.52	12	0.000089	22.24
1.20	31	0.000095	55.89	14	0.000099	26.27

Run on 3.0 GHz Pentium 4 with 256MB of RAM, Chicago regional network: 1778 zones, 12,979 nodes, 39,017 arcs.

5. Conclusion

This paper proposed, justified, and tested a novel traffic assignment algorithm, dubbed Algorithm B, that efficiently moves flow from costliest paths to cheapest paths until all used paths' costs are within ϵ of their cheapest path. Algorithm B's simple approach is intuitive, demonstrably correct, time and space efficient, and converges satisfactorily in a practical application. Algorithm B operationalizes this approach by restricting attention to a relatively few path segments in a series of acyclic sub-networks, called bushes, which carry all and only trips from their specific origin.

Because problems on acyclic networks solve much, much easier and faster than do those for general networks, B achieves practical user-equilibrium flows in less time than is commonly the case—no matter how large or congested the network or how poor the initial solution. Our test results support this conclusion. In two test cases, a 80-arc toy network and a 40,000-arc model for Chicago, B significantly out performed both

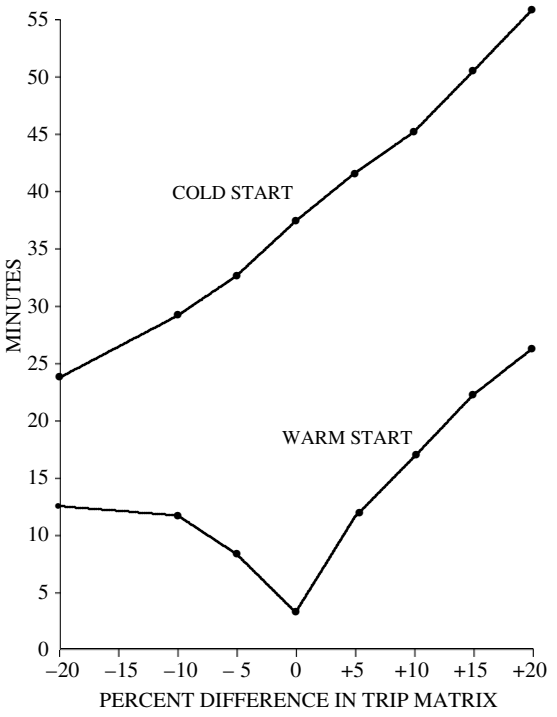


Fig. 13. Time to reach $\gamma < 0.0001$: cold start vs warm start.

the classic LeBlanc–Nguyen Frank–Wolfe algorithm (FW) and the recent Bar-Gera origin-based algorithm (OBA). In both cases, FW was never able to approach the precision B routinely achieves, and OBA always took over five times longer to reach a relative gap below 0.001.

5.1. Future work

Several opportunities present themselves to make B more useful. Its functionality can readily extend in other directions. With trivial modifications, it adapts to finding system optimal equilibria and/or accommodating variable demand. Serendipitously, it is ideal for computing stochastic user-equilibrium for the logit model (Dial, 1971) by facilitating the calculation of path-choice entropy (Akamatsu, 1997). Also, it may apply to dynamic traffic equilibrium (Friesz and Bernstein, 1993), a challenging area always in need of faster algorithms.

5.2. Related prior work

Dafermos and Sparrow (1969) were first to suggest seeking traffic equilibrium by shifting flows from paths with higher costs to those with lower costs, but they provided no practical means of doing so. B's method of shifting flow between independent path segments was originally proposed in Bertsekas (1976). His ideas are implemented in Jayakrishnan et al. (1994); however their algorithm stores paths and makes no use of acyclic networks.

The first published use of potentials-based acyclic networks and topological ordering to obviate path enumeration in traffic assignment appears in Dial (1971). User-equilibrium, acyclic networks and longest paths are applied also to toll-road pricing in Dial (1999a,b). A direct ancestor of this paper is Dial (1999a,b), which proposes using max–min path-cost difference as a termination criterion, and used acyclic networks and path-flow shifting to achieve equilibrium in a network having only one origin. The algorithms reported herein have some similarity to the work of Bar-Gera (1999, 2002) but were developed independently. Furthermore, this

paper provides a superior implementation that results in much more efficient computation than previously reported by any author.

Acknowledgements

Special thanks go to Dr. H. Bar-Gera and the Open Channel Foundation. The latter's web site furnished the network and trip matrix files for the Chicago area, which Caliper Corporation engineers downloaded and passed on to me. It also provided the executable code for Bar-Gera traffic assignment algorithm (OBA) that Caliper used for its testing and I for the performance comparisons reported here.

I am particularly grateful to Caliper's president, Dr. Howard Slavin, for partially funding the R&D this paper describes. It was he who made it possible to return to research I had left five years before with my retirement from the Volpe National Transportation Research Center. My manager there, Mr. Mike Jacobs—now also retired—provided a uniquely productive R&D environment. Without the support of Mssrs. Slavin and Jacobs this paper would not exist. Finally, thanks are due to the generous assistance of two referees and the associate editor.

References

Ahuja, R., Magnanti, T., Orlin, J., 1993. Network Flows. Prentice Hall, Englewood, NJ.

Akamatsu, T., 1997. Decomposition of path choice entropy in general transport networks. Transportation Science 31 (4), 349–362.

Bar-Gera, H., 1999. Origin-based algorithms for transportation network modeling. Technical Report 103. NISS, Research Triangle Park, NC.

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

Bazaraa, M.S., Hanif, D., Shetty, C.M., 1993. NonLinear Programming: Theory and Algorithms. John Wiley & Sons. Inc.

Beckmann, M., McGuire, C., Winston, C., 1956. Studies in the Economics of Transportation. Yale University Press, New Haven, CT.

Bertsekas, D., 1976. On the Goldstein–Levitin–Polyak gradient projection method. IEEE Transactions on Automatic Control 21 (2), 174–183.

Bertsekas, D., 1995. Nonlinear Programming. Athena Scientific, Belmont, MA.

Boyce, D., Ralevic-Dekic, B., Bar-Gera, H., 2004. Convergence of traffic assignments: how much is enough? ASCE Journal of Transportation Engineering (January/February).

Brandon, J., Rabinowicz, A., Slavin, H., 2005. Experiments with alternative traffic assignment methods. Presentation at the 2005 Transportation Research Board's Conference for Planning Applications.

Dafermos, S., Sparrow, R., 1969. The traffic assignment problem for a general network. Journal of Research of the National Bureau of Standards 73B (22), 91–118.

Dantzig, G., 1963. Linear Programming and Extensions. Princeton University Press, Princeton, NJ.

Dial, R., 1971. A probabilistic multipath assignment model that obviates path enumeration. Transportation Research 5, 83–111.

Dial, R., 1999a. Minimal-revenue congestion pricing. Part I: a fast algorithm for the single-origin case. Transportation Research Part B 33, 189–202.

Dial, R., 1999b. Accurate traffic equilibrium: how to bobtail Frank–Wolfe. Technical Report. Volpe National Transportation Research Center, Cambridge, MA.

Federal Highway Administration, 1973. Taffic Assignment. US Government Printing Office, Washington, DC.

Frank, M., Wolfe, P., 1956. An algorithm for quadratic programming. Naval Rsearch Logistics Quarterly 3, 95–110.

Friesz, T., Bernstein, D., Smith, T., Tobin, R., Wie, B., 1993. A variational inequality formulation of the dynamic network user equilibrium problem. Operations Research 41 (1), 179–191.

Horowitz, A., 1991. Delay-Volume Relations for Travel Forecasting, Based on the 1985 Highway Capacity Manual. US Department of Transportation, Federal Highway Administration.

Jayakrishnan, R., Tsai, W., Prashker, J., Rajadhyaksha, S., 1994. A faster path-based algorithm for traffic assignment. Transportation Research Record 1443, 75–83.

LeBlanc, L., 1973. Mathematical programming algorithms for large scale network equilibrium and network design problems. Ph.D. Dissertation. Department of Industrial Engineering and Management Sciences, Northwestern University.

Nguyen, S., 1973. A mathematical programming approach to equilibrium methods of traffic assignment with fixed demands. Publication 138, Département d'Informatique et de Recherche Opérationelle, Université de Montréal, Montreal.

Patriksson, M., 1994. The Traffic Assignment Problem: Models and Methods. VSP, Utrecht, The Netherlands.

Rabinowicz, A., 2005. Personal Correspondence. Caliper Corporation.

Rockafellar, R.T., 1984. Network Flows and Monotropic Optimization. John Wiley & Sons, New York.

Urban Mass Transit Administration, 1977. UMTA Transportation Planning System Reference Manual, U.S. Department of Transportation, Washington, D.C.

Wardrop, J., 1952. Some theoretical aspects of road traffic research. Proceedings of the Institution of Civil Engineers, Part 2, 325–378.