A SLOPE-BASED MULTI-PATH FLOW UPDATE ALGORITHM FOR THE STATIC USER EQUILIBRIUM TRAFFIC ASSIGNMENT PROBLEM

By

Amit Kumar

Purdue University

550 Stadium Mall Drive

West Lafayette, IN 47907

Tel: (765) 237-8328

Email: kumar44@purdue.edu

and

Srinivas Peeta

Purdue University

550 Stadium Mall Drive

West Lafayette, IN 47907-2051

Tel: (765) 494-2209

Fax: (765) 496-7996

Email: peeta@purdue.edu

ABSTRACT

This paper presents a new algorithm for solving the static traffic assignment problem that operates in the space of path flows. It uses a sequential equilibration technique whereby origin-destination (O-D) pairs are equilibrated one at a time iteratively. Labeled the slope-based multi-path algorithm (SMPA), the approach inherits some insights from the gradient projection (GP) algorithm of Jayakrishnan et al., Dial's algorithm B, and the recent GP method of Florian et al. However, the flow update mechanism of the SMPA algorithm is new, with different search directions for paths with higher and lower costs than the average cost for an O-D pair. It uses the slopes of cost functions in such a way as to bring the costs of different paths closer and enable faster convergence. Further, it does not require a line search for finding the move size. Computational experiments using the Sioux Falls and Borman Corridor networks provide insights on the SMPA algorithm and compare its performance relative to the GP algorithm of Florian et al. The results illustrate that the SMPA algorithm has a superior rate of convergence.

INTRODUCTION

The traffic assignment problem (TAP) is a key step of the transportation planning process, and seeks to predict network flows. The most widely used objective for the TAP is user equilibrium (UE), which is based on minimizing the travel time of individual drivers. An equilibrium is reached when no driver can improve his or her travel time unilaterally. Wardrop *(1)* formalized the idea of user equilibrium through his first principle. The UE assignment problem can be formulated as a non-linear optimization problem with linear constraints *(2)*.

Several iterative algorithms have been proposed to solve the TAP, based on more than five decades of active research in this field. New algorithms mostly focus on improving the computational efficiency attainable through existing algorithms. Although these algorithms differ in the way the flows are updated in each iteration, they have many common characteristics. Most start with an initial feasible point, which represents a flow vector that satisfies the constraint set. The iterative process brings them closer to convergence with each successive iteration. As the exact equilibrium point is difficult to reach in finite time, these algorithms use threshold stopping criteria to terminate. As discussed later in the paper, several convergence measures have been proposed in the literature for tracking the threshold criteria for termination.

From a methodological standpoint, existing algorithms differ not only in the way they update the flow vector, but also in terms of whether they update the flow vector for all origin-destination (O-D) pairs simultaneously or sequentially "one-at-a-time". Chen and Jayakrishnan *(3)* suggest that the "one-at-a-time" technique based on the Gauss-Seidal decomposition scheme has better convergence properties because the information obtained from the flow update of one O-D pair can be utilized to determine the flow update direction for the next O-D pair as well as to update the feasible path set. This is not possible when the flow update is performed for all O-D pairs simultaneously. However, the sequential technique has the additional computational burden of requiring a line search for each O-D pair unlike the single line search for the simultaneous update case. But, as it requires fewer iterations to achieve the same level of convergence, the sequential technique generally has better convergence properties *(4)*.

A third aspect in which UE solution algorithms differ is in terms of the number of paths which participate in the equilibration process at a time for each O-D pair. Many algorithms *(5, 6)* which operate in the space of path flows consider just two paths at a time in the equilibration process. These are generally the costliest and cheapest paths in the feasible path set. A portion of flow is transferred from the costliest path to the cheapest path at a time based on the flow update technique of the algorithm. Another class of algorithms *(7, 8)* uses all feasible paths simultaneously for the equilibration process, leading to a multi-path flow update technique. Here, the information from the flow update of one O-D pair is utilized for updating the feasible path set for the next O-D pair. By contrast, the two-path based algorithms typically avoid using flow update information from one path pair for the next path pair of that O-D pair so as to reduce the computational burden that would otherwise result from repeating the update process for each path pair within an O-D pair. Hence, the multi-path flow update technique can be a better approach as it uses the information from each flow update.

This paper presents a new algorithm labeled as the slope-based multi-path algorithm (SMPA) which operates in the space of path flows, uses a sequential O-D equilibration scheme, and updates all feasible paths of an O-D pair simultaneously. The novelty of this algorithm lies in two aspects of the flow update mechanism: (i) obviating the need for a line

search in each iteration, and (ii) the way in which the sensitivity of path costs relative to flow, referred to as slopes, are used in the equilibration process.

The rest of the paper is organized as follows. The next section reviews the literature in user equilibrium assignment and the characteristics of various types of solution algorithms. Then, the UE problem is formulated and the SMPA algorithm is discussed in detail. Next, some practical aspects of the SMPA algorithm, and related parameters, are discussed. This is followed by an illustration of the computational results and sensitivity analysis from numerical experiments using the Sioux Falls and Borman Corridor networks. Finally, some concluding remarks are presented.

LITERATURE REVIEW

The idea of equilibrium in traffic flow was first proposed by the economist Frank Knight in 1924, but was formally defined by John Wardrop in 1952 through what is labeled as Wardrop's first principle *(1)*. A few years later, Beckmann et al. *(9)* proposed the mathematical equivalent of user equilibrium as an optimization problem, which is known as Beckmann's transformation. They not only proved the equivalency between this transformation and user equilibrium but also proved that it has a unique solution in terms of link flows under certain simplifying assumptions. In the same year, a linear approximation method was devised by Frank and Wolfe *(10)* to solve quadratic problems. The Frank-Wolfe (F-W) algorithm was used to solve the UE assignment problem by Bruynooghe et al. *(11),* and later more effectively by LeBlanc et al. *(12, 13)* and Nguyen *(14)* independently. Since then, the F-W algorithm has been the most commonly used method to solve the UE assignment problem in practice.

The key benefits of the F-W algorithm are its low memory requirement and simplicity of execution. However, it has some important drawbacks, the most significant of which is its sub-linear convergence due to which it starts tailing into endless creep near convergence. Its poor performance near convergence is well-known, and is attributed to the fact that its search direction becomes orthogonal to the gradient of the objective function near convergence. Consequently, several studies in the literature have focused on improving the convergence properties of the F-W algorithm by improving the search direction and modifying its move size. This has led to many variants of this algorithm which either modify the search direction or move size or both. One such improvement is the parallel tangent direction originally suggested by Luenberger *(15)* and introduced in the TAP by LeBlanc et al. *(16)*. It was further improved by Florian et al. *(17)* and analytically implemented by Arezki and Van Vliet *(18)*. Another development based on improving search direction was devised by Fukushima *(19)*. His method uses the search directions from previous iterations to find a nominal new search direction as a weighted linear combination. Then, the choice of the new search direction between the nominal search direction computed and the original F-W direction is made by using the directional derivative of the objective function. Hearn et al. *(20)* proposed a similar concept but using only a restricted set of previous solution points of the linearized sub-problem which is termed as the restricted simplicial decomposition algorithm. Weintraub et al. *(21)* suggested a modified move size to reduce the zigzagging nature of the F-W algorithm. Most of these algorithms, obtained from the modified search direction and move size of the F-W algorithm, operate in the space of link flows and update all O-D pairs simultaneously.

Another class of algorithms operates in the space of path flows. The earliest example of such an algorithm comes from the method suggested by Dafermos *(22)* and Dafermos and Sparrow *(23)*. It involves the optimal flow transfer from the costliest path to the cheapest path for an O-D pair at a time. The move size which determines the optimal flow is found by

minimizing the objective function value. This method was not implemented due to its higher memory requirements and the available computational technology at that time. However, the need for path-based solutions by planners and system operators in recent years, synergized by the vastly enhanced computational capabilities over the past two decades, led to a re-visit of the idea proposed by Dafermos and Sparrow. Important initiatives in this context are by Larson and Patriksson *(24)* and Jayakrishnan et al. *(7)*. Larson and Patriksson used the concept of simplicial decomposition for the disaggregate representation of the equilibrium problem, and the associated solution algorithm is labeled as the disaggregate simplicial decomposition (DSD). The DSD methodology iterates between a master problem and a sub-problem. In the initial few iterations, a first order reduced gradient method is used to solve the reduced master problem and obtain a near-optimal solution. In later iterations, a second order diagonalized Newton method is used to determine a highly accurate solution. The solution is in terms of flow proportions which are multiplied by the O-D demand to obtain the path flows. A key advantage of this algorithm is its excellent reoptimization capabilities. For example, for the determination of the solution for a new O-D demand matrix, the flow proportions from the previous O-D demand matrix act as an efficient warm start point. Jayakrishnan et al. *(7)* adopt the basic Goldstein-Levitin-Polyak gradient projection (GP) algorithm (*25, 26)* to solve the assignment problem in the path flow space. The search direction in each iteration is the minimum of the Newton approximation of the transformed objective function. The demand constraints in the original UE formulation are incorporated in the transformed objective function, resulting in a new formulation with only non-negativity path flow constraints. Thereby, if the search direction results in an infeasible flow due to the violation of the non-negativity constraints, a projection is made to the constraint boundary by making the negative flow variable zero.

Bar-Gera *(5, 27)* proposes an origin-based flow proportions approach that is conceptually similar to the algorithms proposed by Gallager *(28)* and Bertsekas *(29)* for routing in telecommunication networks. The main difference is that unlike Bar-Gera the algorithms by Gallager and Bertsekas update the flow using destination-based flow proportions. This algorithm, termed OBA (origin-based algorithm), updates the flow on the origin-based acyclic sub-network. The OBA algorithm considers one origin at a time sequentially to update the origin-based flow proportions for each node using a second order quasi-Newton search direction. The move size is determined by using a heuristic approach that uses the concept of social pressure introduced by Kupsizewska and Van Vliet *(30)*. It is shown that the OBA outperforms the F-W algorithm both in accuracy and computational efficiency.

Dial *(6)* proposes an algorithm labeled algorithm B that works directly in the space of path flows and updates flow by shifting flow from the costlier to the cheaper path. Akin to the OBA algorithm, it uses the concept of origin-based acyclic network. The algorithm involves the transfer of flow between a pair of paths iteratively until their cost difference is within a pre-determined threshold. Then, this procedure is repeated for another pair of paths. When all paths have costs that are within the pre-determined threshold of the cheapest path for an O-D pair, that O-D pair is implied as being equilibrated. The equilibrium process shifts to the next O-D pair for the same origin. After all O-D pairs for an origin are equilibrated, the procedure shifts to the next origin-based sub-network (called a bush). If a new path is found with cost lesser than that of any path in the bush, the bush is updated by including this path. The algorithm terminates when all bushes are equilibrated.

More recently, Florian et al. *(8)* develop an algorithm based on the projected gradient method of Rosen *(31)*. It also operates in the space of path flows and equilibrates one O-D pair at a time sequentially. A key difference is that while the Jayakrishan et al. *(7)* use a constant scaling factor as the move size, Florian et al. *(8)* requires a line search technique to

determine the move size. The flow update mechanism in Florian et al. *(8)* is intuitive and uses the average cost of an O-D pair to determine the search direction. The flows are transferred from the paths having cost greater than the average to the paths with costs lesser than the average cost.

The SMPA algorithm proposed in this paper for the UE assignment problem inherits some insights from Jayakrishnan et al. *(7)*, Dial *(6)* and Florian et al. *(8)* , but its flow update mechanism is new and differs from them. Akin to the GP algorithm of Jayakrishnan et al., the SMPA algorithm operates in the path flow space and uses a projection to the constraint boundary by making negative flow variables zero whenever the minimum in the search direction violates the non-negativity constraints. The difference lies in the search direction; Jayakrishnan et al. use the cost of the cheapest path to determine the search direction while the SMPA uses the average cost for it. Thereby, the SMPA algorithm seeks to move path costs towards the average cost for an O-D pair at each iteration, akin to Florian et al. *(8)*. However unlike Florian et al., it does not require a line search for the move size determination. Further, the SMPA has different flow update mechanisms for the set of paths with higher costs than average as compared to the set of paths with costs lower than average. Finally, the SMPA algorithm inherits the concept of equilibrating an O-D pair before bringing another O-D pair into the equilibration process, similar to Dial's algorithm B. However, it operates on all paths of an O-D pair simultaneously unlike in Dial's algorithm which operates on just a pair of paths at a time.

PROBLEM FORMULATION AND ALGORITHM DEVELOPMENT

Let us consider a network with a set of nodes $\mathcal{N}$, and a set of links A. Let $\mathcal{R}$ be the set of origin nodes and $\mathcal{S}$ be the set of destination nodes which may not be mutually exclusive. The set of paths that connect an O-D pair *r-s* is denoted as K^{rs} and the O-D demand as q_{rs}. Let x_a and c_a represent the flow and cost on link a respectively, and f_k^{rs} and c_k^{rs} represent the flow and cost on path k of O-D pair *r-s*, respectively. The cost of traveling on a path is equal to the sum of the costs on the links in that path, where the link cost is a function $c_a(x_a)$ of the flow on that link. The UE assignment problem can be formulated as an optimization problem by the well-known Beckmann's transformation (*9*):

$$\min \quad f(x) = \sum_{a \epsilon A} \int_0^{x_a} c_a(w) dw \tag{1}$$

Subject to:

$$\sum_k f_k^{rs} = q_{rs}\,, \quad \forall\, rs \quad (flow\ conservation\ constraint) \tag{1a}$$

$$f_k^{rs} \geq 0\,, \quad \forall\, k,\ \forall\, rs \quad (non-negativity\ constraint) \tag{1b}$$

$$x_a = \sum_r \sum_s \sum_k f_k^{rs}\, \delta_{a,k}^{rs}\,, \quad \delta_{a,k}^{rs} = 1\ when\ link\ a\ lies\ on\ path\ k, 0\ otherwise \tag{1c}$$

The equivalency of this formulation to the static UE problem and the uniqueness of the solution in terms of link flows is based on the following: (i) the O-D demand is constant and non-negative for all O-D pairs, (ii) the link costs are positive and the link cost functions are monotonically increasing, continuously differentiable functions of flow, and (iii) a link cost depends only on the flow on that link and does not depend on the flow on other links.

The above formulation is usually solved using an iterative approach. The approach starts with a feasible point and seeks to bring it closer to equilibrium in successive iterations by identifying an optimal search direction and move size. Let us consider an intermediate iteration of the equilibration process, represented by feasible links flows that do not satisfy UE. Here, feasible flows refer to the vector of flows that satisfies the set of constraints (1a, 1b, and 1c) in the UE formulation. As the UE conditions are not satisfied, paths with non-zero flows will not have equal cost for an O-D pair. This implies the existence of paths with non-zero flow but with costs higher than that of the minimum cost path for that O-D pair. Then, the equilibration process will entail the shifting of flows from the costlier to the cheaper paths. An optimal shift from the costlier paths to the cheaper paths will move the resulting network flows towards equilibrium, thereby decreasing the objective function value *(Lemma 2, 6)*. Dial (*6*) proposed a mechanism for the optimal flow transfer from the costlier path to the cheaper path, by taking just one pair of paths at a time. In the SMPA algorithm, we propose a mechanism to equilibrate the entire feasible set of paths belonging to an O-D pair simultaneously. That is, instead of transferring flow from one path (costlier) to another path (cheaper), flow is optimally shifted from the set of paths with higher path costs than average for that O-D pair to the set of paths with lower costs than average, inheriting the idea from the GP algorithm of Florian et al. *(8)*. To enable this in the SMPA, the objective function during the equilibration process for an O-D pair *r-s* can be decomposed into three parts:

$$f(x) = \sum_{a \epsilon \overline{P}^{rs} \setminus \underline{P}^{rs}} \int_0^{x_a - \overline{\Delta x}_a} c_a(w)dw + \sum_{a \epsilon \underline{P}^{rs}} \int_0^{x_a + (\underline{\Delta x}_a - \overline{\Delta x}_a)} c_a(w)dw + \sum_{a \epsilon A \setminus (\underline{P}^{rs} \cup \overline{P}^{rs})} \int_0^{x_a} c_a(w)dw \quad (2)$$

Where,
$\overline{P}^{rs}$ = *set of costlier paths, comprising of paths having cost greater than the average cost for the O-D pair r-s*
$\underline{P}^{rs}$ = *set of cheaper paths, comprising of paths having cost lesser than the average cost for the O-D pair r-s*
$\overline{\Delta x}_a$ = *change in flow for link a due to flow update of paths in the costlier set*
$\underline{\Delta x}_a$ = *change in flow for link a due to flow update of paths in the cheaper set*

The third part in Equation (2) considers links which do not belong to any path in the feasible set for the O-D pair *r-s* being equilibrated. The equilibration process represented by Equation (2) is repeated until all O-D pairs are equilibrated. The superscript *rs* is dropped hereafter for simplicity of notation as we focus on an O-D pair *r-s*.

Path set $\overline{P}$ (set of costlier paths)

For path set $\overline{P}$, we transfer flow such that the new path costs resulting from the new flows bring the path costs towards the average cost across all paths for the O-D pair *r-s*. Using a first order Taylor expansion:

$$\bar{c}_k\left(\bar{f}_k - \Delta\bar{f}_k\right) = \bar{c}_k\left(\bar{f}_k\right) - \bar{s}_k\left(\bar{f}_k\right).\Delta\bar{f}_k = c_{av}\,, \quad \forall k \in \overline{P} \in K \quad (3)$$

Where,
$\bar{c}_k$ = *cost of path k belonging to the set* $\overline{P}$

$\bar{f}_k$ = *flow of path k belonging to the set* $\overline{P}$
$\bar{s}_k$ = *first derivative of the cost function of path k belonging to the set* $\overline{P}$
c_{av} = *average cost obtained by averaging the feasible path costs for the O-D pair*

From Equation (3) we obtain:

$$\Delta\bar{f}_k = \frac{\bar{c}_k(\bar{f}_k) - c_{av}}{\bar{s}_k(\bar{f}_k)} \tag{4}$$

Path set $\underline{P}$ (set of cheaper paths)

For path set $\underline{P}$, we add flow to the paths such that the new path costs resulting from the new flows bring these costs to a higher value μ. This new cost μ may not be equal to average path cost c_{av} for O-D pair *r-s*, but will definitely be close to it. Using a first order Taylor expansion:

$$\underline{c}_l\left(\underline{f}_l + \Delta\underline{f}_l\right) = \underline{c}_l\left(\underline{f}_l\right) + \underline{s}_l\left(\underline{f}_l\right).\Delta\underline{f}_l = \mu, \qquad \forall l \in \underline{P} \in K \tag{5}$$

Where,
$\underline{c}_l$ = *cost of path l belonging to the set* $\underline{P}$
$\underline{f}_l$ = *flow of path l belonging to the set* $\underline{P}$
$\underline{s}_l$ = *first derivative of the cost function of path l belonging to the set* $\underline{P}$

From Equation (5) we obtain:

$$\Delta\underline{f}_l = \frac{\mu - \underline{c}_l\left(\underline{f}_l\right)}{\underline{s}_l\left(\underline{f}_l\right)} \tag{6}$$

For flow conservation for the O-D pair, the total flow transferred from the various paths from set $\overline{P}$ should be equal to the sum of flows being added to the paths of set $\underline{P}$. Hence,

$$\sum_{k\epsilon\overline{P}} \Delta\bar{f}_k = \sum_{l\epsilon\underline{P}} \Delta\underline{f}_l \tag{7}$$

Using Equations (4), (6) and (7) we obtain:

$$\sum_{k\epsilon\overline{P}} \frac{\bar{c}_k(\bar{f}_k) - c_{av}}{\bar{s}_k(\bar{f}_k)} = \sum_{l\epsilon\underline{P}} \frac{\mu - \underline{c}_l\left(\underline{f}_l\right)}{\underline{s}_l\left(\underline{f}_l\right)}$$

$$\Rightarrow \sum_{k\epsilon\overline{P}} \frac{\bar{c}_k(\bar{f}_k)}{\bar{s}_k(\bar{f}_k)} - \sum_{k\epsilon\overline{P}} \frac{c_{av}}{\bar{s}_k(\bar{f}_k)} = \mu \sum_{l\epsilon\underline{P}} \frac{1}{\underline{s}_l\left(\underline{f}_l\right)} - \sum_{l\epsilon\underline{P}} \frac{\underline{c}_l\left(\underline{f}_l\right)}{\underline{s}_l\left(\underline{f}_l\right)}$$

$$\Longrightarrow \mu = \frac{\sum_{k\epsilon\overline{P}} \frac{\bar{c}_k\left(\bar{f}_k\right)}{\bar{s}_k\left(\bar{f}_k\right)} - \sum_{k\epsilon\overline{P}} \frac{c_{av}}{\bar{s}_k\left(\bar{f}_k\right)} + \sum_{l\epsilon\underline{P}} \frac{\underline{c}_l\left(\underline{f}_l\right)}{\underline{s}_l\left(\underline{f}_l\right)}}{\sum_{l\epsilon\underline{P}} \frac{1}{\underline{s}_l\left(\underline{f}_l\right)}}$$

From Equation (6) we have:

$$\Delta\underline{f}_l = \frac{\mu - \underline{c}_l\left(\underline{f}_l\right)}{\underline{s}_l\left(\underline{f}_l\right)} = \frac{\mu}{\underline{s}_l\left(\underline{f}_l\right)} - \frac{\underline{c}_l\left(\underline{f}_l\right)}{\underline{s}_l\left(\underline{f}_l\right)}$$

Substituting for μ,

$$\Delta\underline{f}_l = \frac{\sum_{k\epsilon\overline{P}} \frac{\bar{c}_k\left(\bar{f}_k\right)}{\bar{s}_k\left(\bar{f}_k\right)} - \sum_{k\epsilon\overline{P}} \frac{c_{av}}{\bar{s}_k\left(\bar{f}_k\right)} + \sum_{l\epsilon\underline{P}} \frac{\underline{c}_l\left(\underline{f}_l\right)}{\underline{s}_l\left(\underline{f}_l\right)}}{\underline{s}_l\left(\underline{f}_l\right) \cdot \sum_{l\epsilon\underline{P}} \frac{1}{\underline{s}_l\left(\underline{f}_l\right)}} - \frac{\underline{c}_l\left(\underline{f}_l\right)}{\underline{s}_l\left(\underline{f}_l\right)}$$

We now have the following flow update mechanism:

$$\bar{f}_k \rightarrow \bar{f}_k - \Delta\bar{f}_k \ , \qquad k \in \overline{P} \qquad (A)$$

$$\underline{f}_l \rightarrow \underline{f}_l + \Delta\underline{f}_l \ , \qquad l \in \underline{P} \qquad (B)$$

Expressions (A) and (B) are used iteratively until the maximum difference between the costs of used paths (that is, those with non-zero flow) is less than a predefined threshold value β. While transferring flow at each move, the maximum amount of flow which can be transferred from any path is at most equal to the current flow on that path to satisfy the non-negativity constraint. Hence, the $\Delta\bar{f}_k$ and $\Delta\underline{f}_l$ to be used at each move are:

$$\Delta\bar{f}_k = \min\left\{\bar{f}_k, \frac{\bar{c}_k\left(\bar{f}_k\right) - c_{av}}{\bar{s}_k\left(\bar{f}_k\right)}\right\} \qquad (8)$$

$$\Delta\underline{f}_l = \frac{\sum_{k\epsilon\overline{P}} \min\left\{\bar{f}_k, \frac{\bar{c}_k\left(\bar{f}_k\right) - c_{av}}{\bar{s}_k\left(\bar{f}_k\right)}\right\} + \sum_{l\epsilon\underline{P}} \frac{\underline{c}_l\left(\underline{f}_l\right)}{\underline{s}_l\left(\underline{f}_l\right)}}{\underline{s}_l\left(\underline{f}_l\right) \cdot \sum_{l\epsilon\underline{P}} \frac{1}{\underline{s}_l\left(\underline{f}_l\right)}} - \frac{\underline{c}_l\left(\underline{f}_l\right)}{\underline{s}_l\left(\underline{f}_l\right)} \qquad (9)$$

Figure 1 illustrates the logic of the proposed flow update mechanism. For example, let us assume that an O-D pair has six paths in the feasible set of paths labeled 1 to 6 in the figure. The paths labeled 1, 2 and 3 have costs lesser than the average cost and will form the cheaper path set. The paths labeled 4, 5 and 6 have costs greater than the average cost and will form the costlier path set. At each move, the flow update mechanism seeks to reduce the costs of costlier paths and bring them to the average cost (c_{av}) for the O-D pair, and aims to

increase the costs of the cheaper paths to a value μ. The positions of the paths after the flow update are represented by the dotted ellipses in the figure. After each move, the flows, the costs and the slopes of links and paths are updated. The new average path cost for the O-D pair is found based on the updated flows. This process is repeated iteratively whereby the difference between c_{av} and μ decreases in each successive iteration. The O-D pair is assumed to be equilibrated if the difference between its μ and c_{av} is less than a pre-specified small threshold value β.

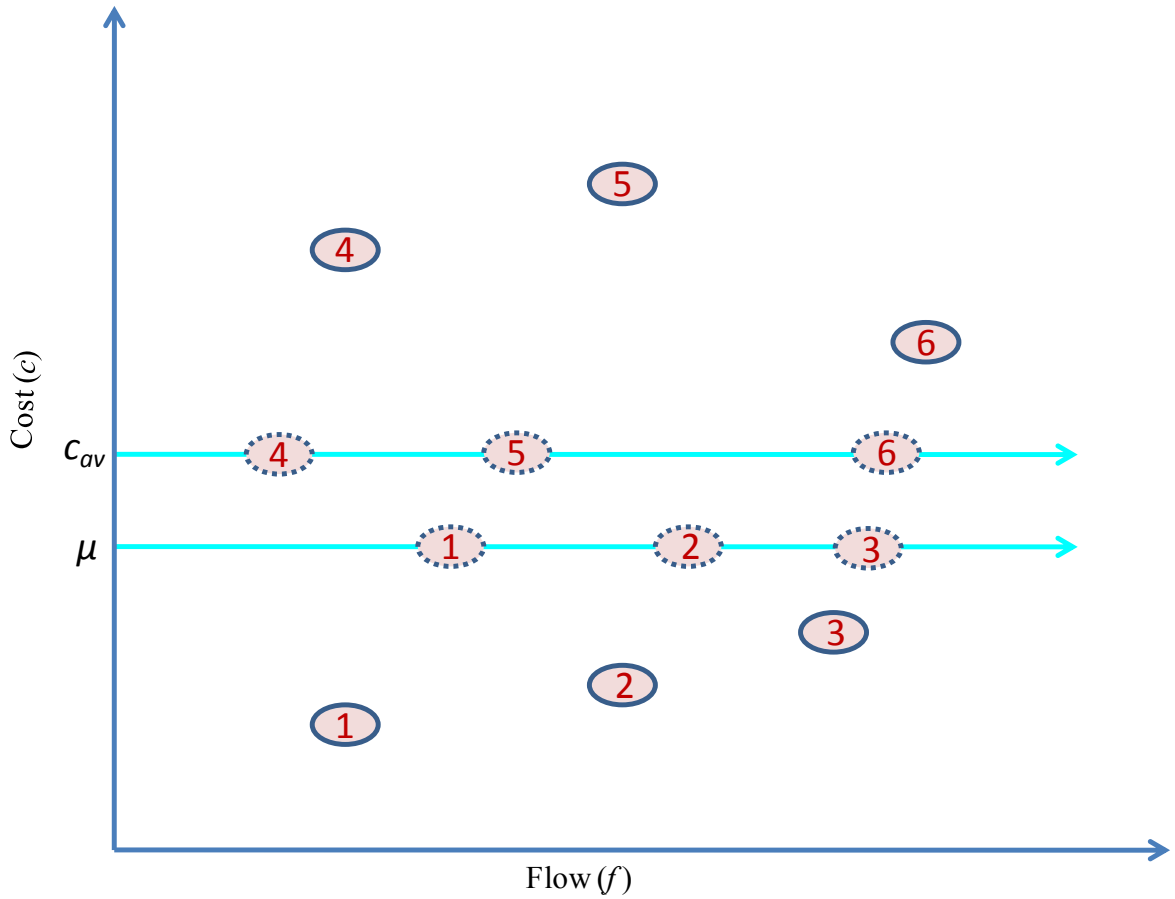

FIGURE 1 The logic of the flow update mechanism

The use of the slopes of the path cost functions represents an important aspect of the SMPA algorithm. The sensitivity of path costs with respect to flow in the form of the slopes of the cost functions are used in the flow update mechanism to determine the optimal flow transfer which brings the costs of different paths in the feasible set closer. The slopes of the path cost functions are derived from the slopes of the link cost functions. However, the slopes of the link cost functions are not constant and depend on the current link flow values. Hence, the slopes of the link cost functions and then those of the paths are updated after each move.

Once an O-D pair is equilibrated, the next O-D pair is brought into the equilibration process in a sequential manner. Before commencing the flow shifts (move) for an O-D pair, if any new path is found that is not in the feasible set and has a lower cost than the average cost for that O-D pair, it is included in its feasible path set. Hence, the SMPA algorithm consists of an inner loop which seeks the equilibration of an O-D pair and an outer loop that sequentially moves from one O-D pair to the next and checks some termination criteria after all O-D pairs are considered. The steps of the SMPA algorithm are illustrated in Figure 2, and the algorithm is summarized thereafter.

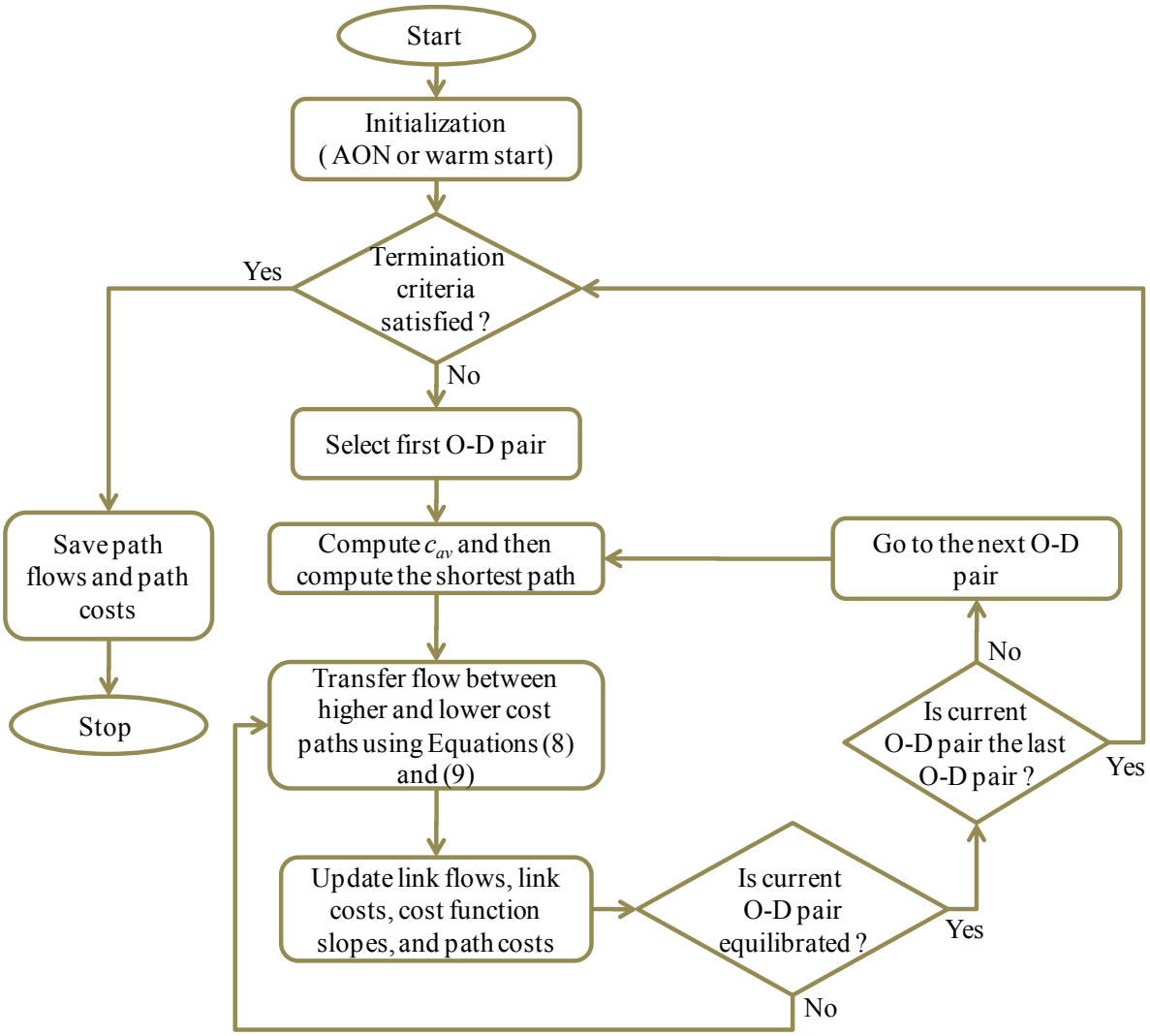

FIGURE 2 The SMPA algorithm.

Steps of the SMPA algorithm

Step 1: Initialize the network using an all-or-nothing (AON) assignment or a warm start. Update the link flows, link costs, slopes of cost functions, and path costs.

Step 2: Check the termination criteria. If termination criteria are satisfied, then stop; the UE solution is the set of the path flows and path costs. Else, go to Step 3.

Step 3: Select the first O-D pair.

Step 4: Find the average cost (c_{av}) of paths in the feasible set (having non-zero flows) for the O-D pair. Then, find the shortest path. If this path is not present in the set of feasible paths and its cost is less than c_{av}, include it in the set of feasible paths and update c_{av}. Else, go to Step 5.

Step 5: Update flows for paths having cost greater than c_{av} using Equation (8) and (*A*).

Step 6: Update flows for paths having cost lower than c_{av} using Equation (9) and (*B*).

Step 7: Update link flows, link costs, slopes of cost functions, and path costs.

Step 8: If the current O-D pair is equilibrated, go to Step 9. Else, go to Step 4.

Step 9: If this is the last O-D pair, go to Step 2. Else, select next O-D pair and go to Step 4.

Analysis of Convergence

The flow vector which solves the minimization problem represented by Beckmann's transformation (Equations 1, 1a, 1b, and 1c) also satisfies the user equilibrium conditions *(2)*. This implies that the minimum of the objective function (1) is obtained when all used paths between an O-D pair have equal (and lowest) cost. Since the decomposed objective function (4) is equivalent to the objective function (1), its minimum will also be achieved when all paths in the feasible set have equal cost. In the decomposed objective function, the links that belong to the third part do not participate in the equilibration process. Hence, the minimum of *f(x)* is obtained when the sum of the first two terms is minimum. As the true equilibrium for real networks is difficult to reach, the equilibrium is assumed to be reached when the difference between path costs in the feasible set is less than a pre-specified threshold β. The equilibration process represented by the flow update using Expressions (A) and (B) iteratively continues until the difference between c_{av} and μ becomes less than β, at which point it is terminated. Then, to prove convergence, it is sufficient to prove that the flow update mechanism decreases the value of the objective function.

Proof by contradiction: Let us suppose that the flow update process increases the value of the objective function at the termination of the equilibration process. This would imply that the increase in the second part of the objective function is larger than the decrease in first part. This implies that the value of μ is higher than the average cost c_{av}; that is, the increase to the objective function value due to the flow increases in the cheaper path set is more than the decrease to the objective function value due to the flow decreases in the costlier path set. This indicates that the equilibration process is not complete. This, along with the increase in the objective function value, implies that the path costs are not close, which means the difference between the μ and c_{av} is greater than β. This contradicts the initial assumption that the equilibration process terminated. This completes the proof.

PRACTICAL ASPECTS OF THE ALGORITHM

The description of the SMPA algorithm in the previous section seems to suggest that it satisfies all constraints in each iteration. However, a careful inspection of the move direction for the cheaper paths indicates the possibility of a theoretical violation of the non-negativity constraint. This situation may arise when the second term in equation (9) is greater than the first term, resulting in a negative flow change for the cheaper path set. Then, if this flow change is greater than the existing path flow, it will result in a negative path flow for that path. To visualize this, consider a situation in which the move direction is such that the value of μ is below the average cost c_{av} and the feasible set has a path with cost smaller than the average cost but greater than μ. Since the SMPA move direction tends to bring all paths having cost lesser than the average cost towards μ, if a path cost is greater than μ, the move direction will seek to reduce it by decreasing its flow. While flow conservation is satisfied in this process, the possibility exists that this move may result in an infeasible flow, thereby violating the non-negativity constraint. While such a situation is not common and will involve small amounts of flow, it is important to resolve this theoretical issue to make the flow update mechanism consistent with the formulation constraints. This is done by the backtracking in the direction opposite to the move direction for paths in the cheaper path set up to the point at which the sum of the flow changes becomes equal to the magnitude of the negative flow. Thereby, the infeasible flow is projected back to the feasible space, satisfying

the non-negativity constraint and the conservation of flow. The detailed sequence of the steps to implement the SMPA algorithm is illustrated in Figure 3.

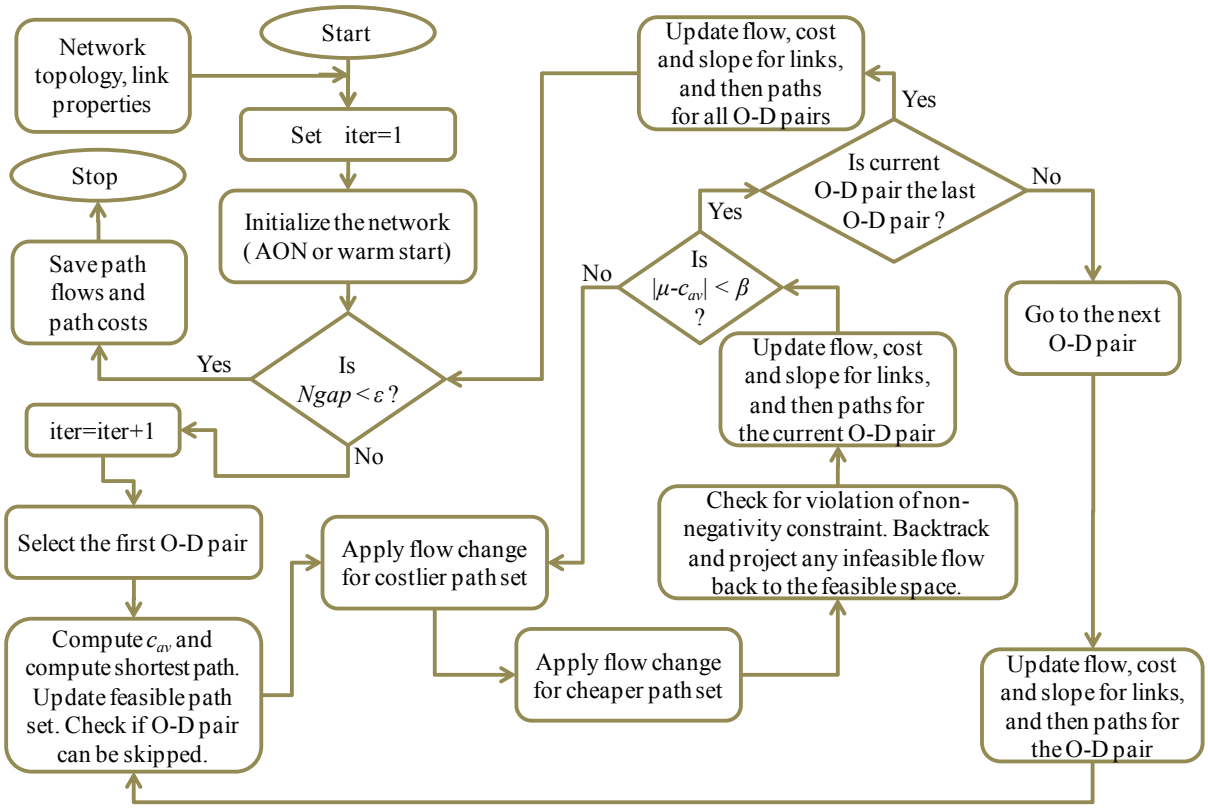

FIGURE 3 Detailed implementation flow chart for the SMPA algorithm.

The second aspect which needs discussion is the interdependencies among path costs. Some paths in the feasible set may have some common links; then, the increase or decrease of flow on one path will alter the cost on the other path(s). The SMPA addresses this problem by incorporating slope (of path cost) terms in the move direction. As the slope of a path cost is obtained by summing the slopes of the associated link cost functions, if a link appears in more than one path the slope of that link will appear the corresponding number of times in the flow update equation of the cheaper path set thereby decreasing the magnitude of the move direction.

The third aspect is the notion of introducing a scaling factor α. The scaling factor here refers to a parameter which is used to scale the move size. It is used to speed up the convergence, and for the SMPA it can be any positive number which remains constant for all iterations. A scaling factor is introduced in the move direction for the costlier path set. As the move direction for the cheaper path set is derived by satisfying the flow conservation constraint (Equation7), it is automatically introduced in its expression. The move directions after introducing the scaling factor are shown in Equations (10) and (11); they replace Equations (8) and (9) during the implementation of the SMPA algorithm.

$$\Delta \bar{f}_k = \min\left\{\bar{f}_k, \alpha \frac{\bar{c}_k(\bar{f}_k) - c_{av}}{\bar{s}_k(\bar{f}_k)}\right\} \tag{10}$$

$$\Delta \underline{f}_l = \frac{\sum_{k \epsilon \bar{P}} \min\left\{\bar{f}_k, \alpha \frac{\bar{c}_k(\bar{f}_k) - c_{av}}{\bar{s}_k(\bar{f}_k)}\right\} + \sum_{l \epsilon \underline{P}} \frac{\underline{c}_l\left(\underline{f}_l\right)}{\underline{s}_l\left(\underline{f}_l\right)}}{\underline{s}_l\left(\underline{f}_l\right) \cdot \sum_{l \epsilon \underline{P}} \frac{1}{\underline{s}_l\left(\underline{f}_l\right)}} - \frac{\underline{c}_l\left(\underline{f}_l\right)}{\underline{s}_l\left(\underline{f}_l\right)} \quad (11)$$

Numerical experiments were performed to obtain the optimum value of the scaling factor α for the test networks. The CPU time to reach the convergence level of 10^{-5} in terms of the termination criterion (normalized gap) was determined for different values of the scaling factor. Figure 4 indicates that the algorithm performs best under the scaling factor 1.5 for Sioux Falls network, which is used for the computational experiments in this paper. A similar approach for the Borman Corridor network indicates that the algorithm performs best under the scaling factor 1.8.

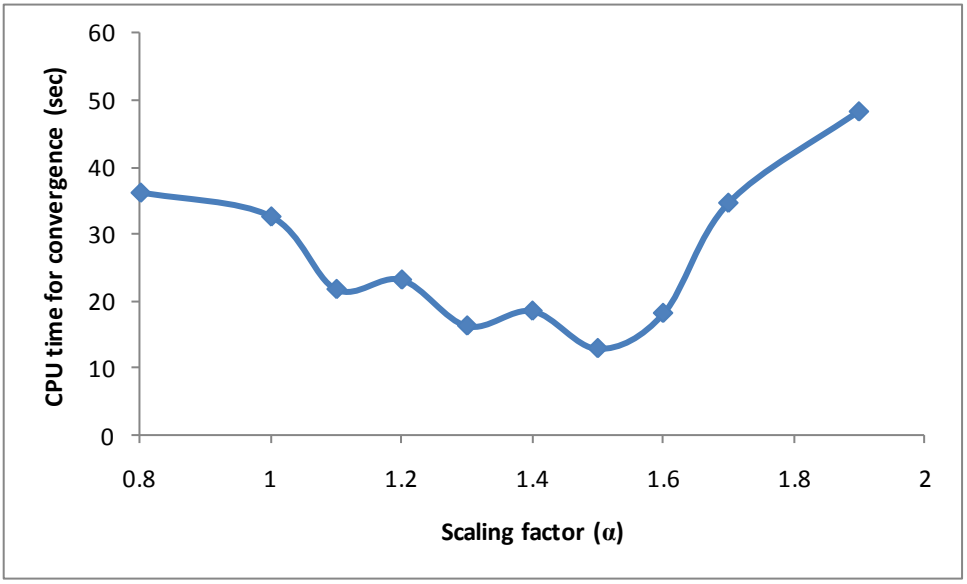

FIGURE 4 Effect of scaling factor on convergence for Sioux Falls network.

The fourth aspect in the implementation context is the measure of convergence, labeled the termination criterion. Past studies have proposed several measures of convergence: decrease in the UE objective function, relative gap, relative lower bound gap, normalized gap, etc. *(32)*. This paper uses the normalized gap (*Ngap*) as the measure of convergence:

$$Ngap = \frac{\left(\sum_r \sum_s \sum_k c_k^{rs} f_k^{rs} - \sum_r \sum_s \sum_k c_{min}^{rs} f_k^{rs}\right)}{\sum_r \sum_s \sum_k f_k^{rs}}$$

The normalized gap, also labeled as the average excess cost, represents the excess cost of paths above the minimum path cost for the O-D pair averaged over the whole network. Another measure, the relative gap (*Rgap*) is defined as:

$$Rgap = \frac{\left(\sum_r \sum_s \sum_k c_k^{rs} f_k^{rs} - \sum_r \sum_s \sum_k c_{min}^{rs} f_k^{rs}\right)}{\sum_r \sum_s \sum_k c_k^{rs} f_k^{rs}}$$

From these definitions, it can be noted that the *Rgap* typically will be smaller than the *Ngap* for the same level of convergence. Boyce et al. *(33)* suggest that an *Rgap* of the order of 10^{-4} may be sufficient for most planning purposes. This paper adopts an *Ngap* of 10^{-5} as the termination criterion for the algorithm.

COMPUTATIONAL RESULTS

The SMPA algorithm was coded in Matlab. The computational environment used is a Dell desktop PC with Intel core 2 quad processors (3 GHz) with 4 GB RAM and 32-bit Windows Vista operating system. The computational experiments were conducted for the Sioux Falls and Borman Corridor networks. The Sioux Falls network consists of 24 nodes, 76 links and 576 O-D pairs and Borman Corridor network in northwest Indiana consists of 197 nodes, 460 links and 1849 O-D pairs. The new GP algorithm (labeled FGP hereafter) of Florian et al. *(8)* was also coded in Matlab using the same data structure and executed on the same computer.

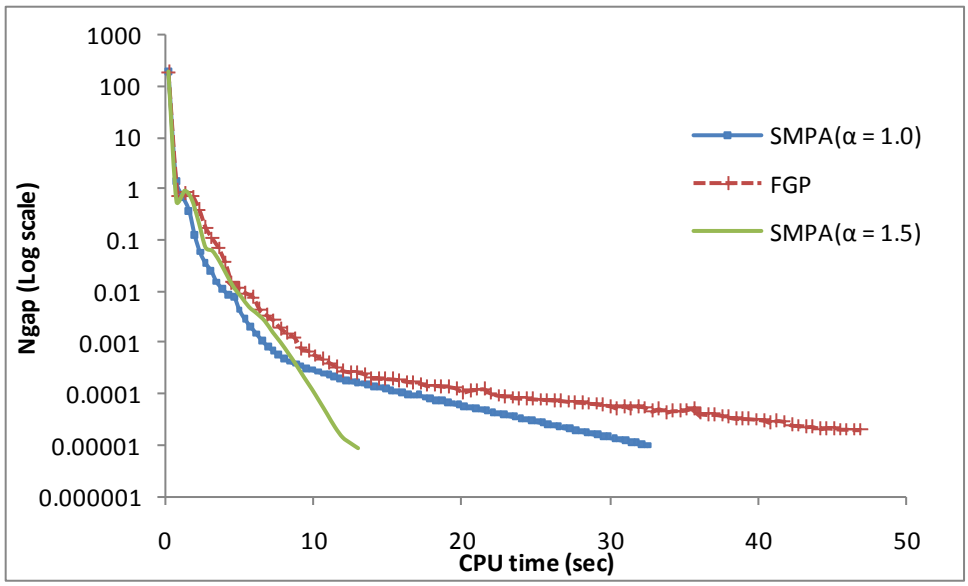

FIGURE 5a Convergence characteristics of the algorithms for the Sioux Falls network.

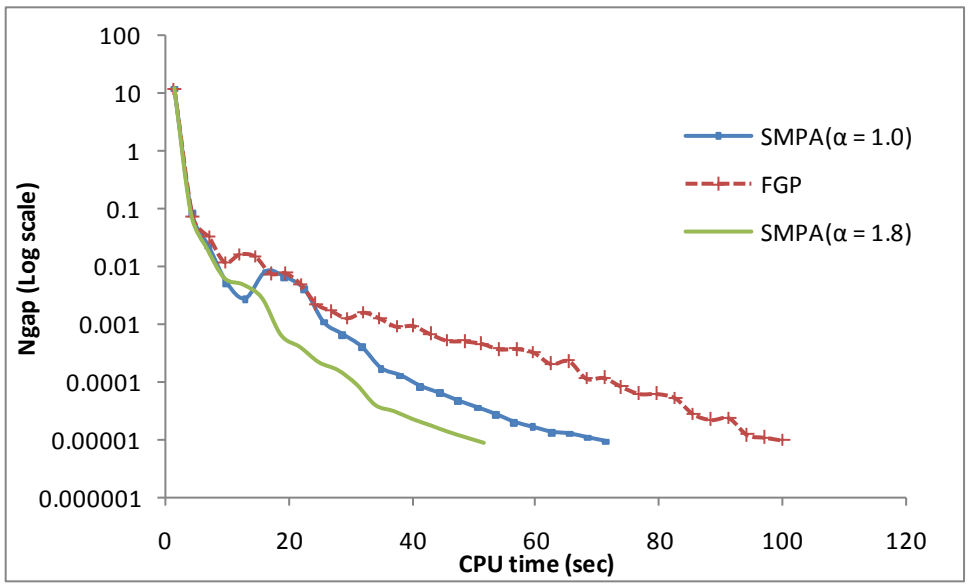

FIGURE 5b Convergence characteristics of the algorithms for the Borman network.

The computational results are shown in Figure 5. The SMPA algorithm performs better under the scaling factor 1.0 in the initial stages, but much better with the scaling factor 1.5 for Sioux Falls network and scaling factor 1.8 for Borman Corridor network in the later stages. Even under the scaling factor 1.0, the SMPA overtakes the FGP in the early stages and consistently performs better until convergence. The FGP was found to be efficient up to the *Ngap* of the order of 10^{-3}, but thereafter the relative performance of SMPA was found to be much better, especially under the scaling factor 1.5. The better performance of SMPA over FGP can be attributed to the efficient utilization of the sensitivity of path costs with respect to the path flows (slopes of cost functions). The FGP uses the same move size for the move direction for all paths of an O-D pair; however, some paths may be more sensitive to flow and their cost function slopes may be higher than those of other paths. By contrast, the SMPA uses the slope of the cost function in the equilibration process in a manner which ensures smaller flow changes for paths with higher slopes of the cost functions.

Sensitivity analysis was performed for both the SMPA and FGP algorithms with respect to the level of demand, and their performance was compared using the Sioux Falls network. The level of demand was varied by multiplying the Sioux Falls demand by factors ranging from 0.8 to 1.2 in increments of 0.2. The scaling factor 1.5 was adopted for the SMPA algorithm for the sensitivity analysis. Figure 6 shows the results under the various demand levels. They indicate that as the level of demand increases the CPU time required to reach the same level of convergence increases for both algorithms. However, the SMPA algorithm performed better in all cases, and by substantial margins as demand increases.

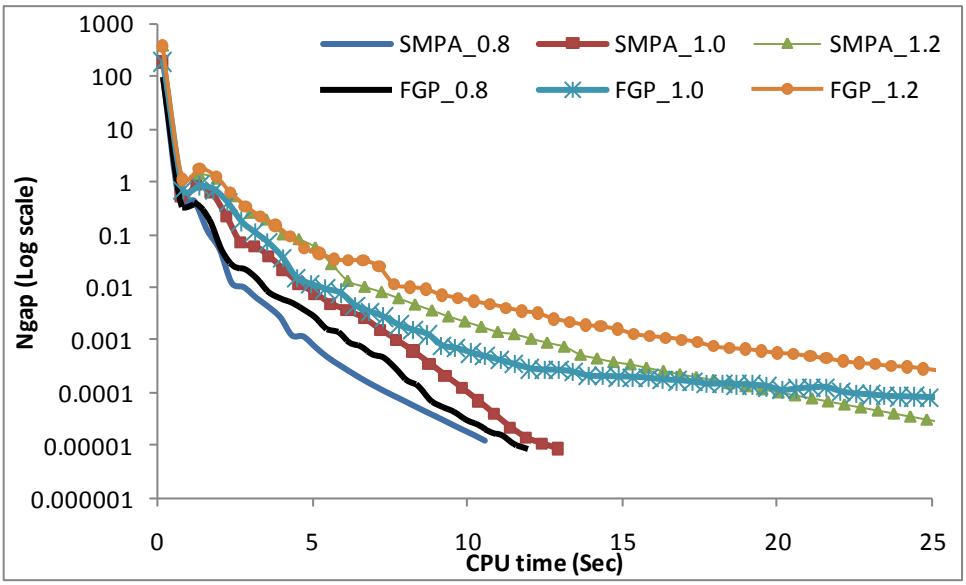

FIGURE 6 Sensitivity analysis for varying demand levels for the Sioux Falls network.

CONCLUDING COMMENTS

This study proposes a new algorithm labeled SMPA for solving the static user equilibrium traffic assignment problem. The algorithm operates in the space of path flows and equilibrates one O-D at a time sequentially. The algorithm uses the sensitivity of path costs relative to flow in the form of the slopes of path cost functions to determine the move directions. The slopes of path costs are used in such a way as to bring the path costs closer at each move; this represents the key aspect of the algorithm in achieving faster convergence. In addition, the SMPA algorithm has different flow update mechanisms for the set of paths with

larger costs than average as compared to the set of paths with costs lower than average. The SMPA algorithm does not use a line search for finding the move size, but uses a constant scaling factor.

Computational experiments were performed using the Sioux Falls and Borman Corridor networks. The SMPA algorithm was benchmarked against the recent GP algorithm of Florian et al. The computational results illustrate that the SMPA algorithm has a faster convergence rate and can be used to obtain the UE solution with high convergence levels. The sensitivity analysis indicates that as the level of demand increases, the performance of both algorithms (FGP and SMPA) diminishes. This may be due to the larger number of feasible paths at higher demand levels which increases the burden of flow update in each iteration. It is possible that the SMPA algorithm may converge even faster if the scaling factor is determined through a line search, but the tradeoffs in terms of the additional computational burden need to be investigated. The calculation of the slopes of the cost functions introduces computational cost in each iteration of the SMPA algorithm, but the gains in terms of faster convergence more than compensate for this burden. Future work will explore computational coding aspects to analyze mechanisms to further improve the performance of this algorithm.

ACKNOWLEDGMENTS

This study was partly funded by a project through the NEXTRANS Center at Purdue University. We would like to acknowledge the beneficial comments from Dr. Hillel Bar-Gera during the algorithm development process. We would also like to thank Dr Lili Du and Salvador Hernández for their feedback during the Matlab coding.

REFERENCES

1. Wardrop, J. Some Theoretical Aspects of Road Traffic Research. *Proceedings of the Institution of Civil Engineers*, Part 2, 1952, pp. 325-378.

2. Sheffi, Y. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

3. Chen, A., and R. Jayakrishnan. A Path-Based Gradient Projection Algorithm: Effects of Equilibration with a Restricted Path Set under Two Flow Update Policies. Presented at 77th Annual Meeting of the Transportation Research Board, Washington, D.C., 1998.

4. Chen, A., R. Jayakrishnan, and W. K. Tsai. Faster Frank-Wolfe Traffic Assignment with New Flow Update Scheme. *Journal of Transportation Engineering,* Vol. 128, No.1, ASCE, 2002, pp. 31-39.

5. Bar-Gera, H. *Origin-Based Algorithms for Transportation Network Modeling*. PhD Dissertation, University of Illinois at Chicago. Technical Repo. No. 103, 1999.

6. Dial, R.B. A Path-Based User-Equilibrium Traffic Assignment Algorithm That Obviates Path Storage and Enumeration. *Transportation Research Part B,* Vol. 40, No. 10, 2006, pp. 917-936.

7. Jayakrishnan, R., W. K. Tsai, J. Prashker, and S. Rajadhyaksha. (1994). A Faster Path-Based Algorithm for Traffic Assignment. *Transportation Research Record: Journal of the Transportation Research Board,* No. 1443, Transportation Research Board of the National Academics, Washington, D.C., 1994, pp. 75-83.

8. Florian, M., I. Constantin, and D. Florian. A New Look at the Projected Gradient Method for Equilibrium Assignment. CD-ROM. Transportation Research Board of the National Academics, Washington, D.C., 2009.

9. Beckmann, M., C. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation.* Yale University Press, New Haven, 1956.

10. Frank, M., and P. Wolfe. An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly*, Vol. 3, 1956, pp. 95-110.

11. Bruynooghe M., A. Gilbert, and M. Sakarovitch. Une Methode d'affectation du Traffic. In: Leutzbach W, Baron P (eds). *Proceedings of the 4th International Symposium on the Theory of Road Traffic.* Bundesminister fur Verkehr, Abt. Strassenbau, Bonn, Germany, 1969.

12. LeBlanc, L. J. *Mathematical Programming Algorithms for Large Scale Network Equilibrium and Network Design Problems*. PhD Dissertation, Department of Industrial Engineering and Management Science, Northwestern University, USA, 1973.

13. LeBlanc, L. J., E. K. Morlok, and W. P. Pierskalla. An Efficient Approach to Solving the Road Network Equilibrium Traffic Assignment Problem. *Transportation Research*, Vol. 9, 1975, pp. 309-318.

14. Nguyen, S. An Algorithm for the Traffic Assignment Problem. *Transportation Science,* Vol. 8, No. 3, 1974, pp. 203-216.

15. Luenberger, D. G. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, Reading, Massachusetts, 1973.

16. LeBlanc, L. J., R. V. Helgason, and D. E. Boyce. Improved Efficiency of the Frank-Wolfe Algorithm for Convex Network Problems. *Transportation Science,* Vol. 19, No. 4, 1985, pp. 445-462.

17. Florian, M., J. Guelat, and H. Spiess. An Efficient Implementation of the "PARTAN" Variant of the Linear Approximation Method for the Network Equilibrium Problem. *Networks,* Vol. 17, 1987, pp. 319-339.

18. Arezki, Y., and D. Van Vliet. A Full Analytical Implementation of the PARTAN/Frank-Wolfe Algorithm for Equilibrium Assignment. *Transportation Science,* Vol. 24, 1990, pp. 58-62.

19. Fukushima, M. A Modified Frank-Wolfe Algorithm for Solving the Traffic Assignment Problem. *Transportation Research Part B*, Vol.18, No. 2, 1984, pp. 169-177.

20. Hearn, D.W., S. Lawphongpanich, and J. A. Ventura. Finiteness in Restricted Simplicial Decomposition. *Operations Research Letters,* Vol. 4, No. 3, 1985, pp. 125-130.

21. Weintraub, A., C. Ortiz, and J. Gonzales. Accelerating Convergence of the Frank-Wolfe Algorithm. *Transportation Research Part B,* Vol. 19, No. 2, 1985, pp. 113-122.

22. Dafermos, S. *Traffic Assignment and Resource Allocation in Transportation Networks*. PhD Dissertation, Johns Hopkins University, Baltimore, MD, 1968.

23. Dafermos, S., and R. Sparrow. The Traffic Assignment Problem for a General Network. *Journal of Research of the National Bureau of Standards,* Vol. 73B, 1969, pp. 91-118.

24. Larsson, T., and M. Patriksson. Simplicial Decomposition with Disaggregate Representation for the Traffic Assignment Problem. *Transportation Science,* Vol. 26, No. 1, 1992, pp. 4-17.

25. Bertsekas, D.P. On the Goldstein-Levitin-Poljak Gradient Projection Method. *IEEE Transactions on Automatic Control*, Vol. 21, No. 2, 1976, pp. 174-184.

26. Bertsekas, D.P., and R. G. Gallager. *Data Networks.* Englewood Cliffs, Prentice-Hall, NJ, 1987.

27. Bar-Gera, H. Origin-Based Algorithm for the Traffic Assignment Problem. *Transportation Science,* Vol. 36, No. 4, 2002, pp. 398-417.

28. Gallager, R. G. A Minimum Delay Routing Algorithm Using Distributed Computation. *IEEE Transactions on Communications*, Vol. 25, No. 1, 1977, pp. 73-85.

29. Bertsekas, D. P. Algorithms for Nonlinear Multicommodity Network Flow Problems. *In Proceedings of the International Symposium on Systems Optimization and Analysis*, A. Bensoussan and J. L. Lions, eds., Springer-Verlag, Berlin,1979, pp. 210-224.

30. Kupsizewska, D., and D. Van Vliet. 101 Uses for Path-Based Assignment. *Transport Planning Methods: Proc., Seminar C, PTRC Planning and Transport Summer Ann. Meeting*, Univ. of Sussex, U.K., 1999.

31. Rosen, B. The Gradient Projection Method for Nonlinear Programming, part I. Linear Constraints. *Journal of the Society for Industrial and Applied Mathematics,* Vol. 8, No. 1, 1960, pp. 181-217.

32. Rose, G., M. S. Daskin, and F. S. Koppelman. An Examination of Convergence Error in Equilibrium Traffic Assignment Models. *Transportation Research part B,* Vol. 22, No. 4, 1988, pp. 261-274.

33. Boyce, D., B. Ralevic-Dekic, and H. Bar-Gera. Convergence of Traffic Assignments: How Much is Enough? *Journal of Transportation Engineering,* Vol. 130, No. 1, ASCE, 2004, pp. 49-55.