

A Link-Node Complementarity Formulation and Its Solution Algorithm for Asymmetric Traffic Assignment

Jeff X. Ban*

California Center for Innovative Transportation (CCIT)
University of California - Berkeley
2105 Bancroft Way, Suite 300
Berkeley, CA 94720-383
Phone: 510-642-5112
Fax: 510-642-0910
Email: xban@calccit.org
* **Corresponding Author**

Henry X. Liu

Department of Civil Engineering
University of Minnesota – Twin Cities
Tel: (612) 625-6347, Fax: (612) 626-7750
Email: henryliu@umn.edu

Michael C. Ferris

Computer Sciences Department
University of Wisconsin at Madison
Tel: (608) 262-4281, Fax: (608) 262-9777
Email: ferris@cs.wisc.edu

Re-submittal to 85th Transportation Research Board Annual Meeting

November 15, 2005

Word Counts: 5,668 + 1,750 (1 table and 6 figures) = 7,418 words

Abstract

This paper presents a complementarity formulation and its solution algorithm for asymmetric traffic assignment. A link-node based nonlinear complementarity formulation is first proposed for modeling the asymmetric user equilibrium, with rigorous proofs of equivalence, and solution existence and uniqueness conditions. To solve the model, we develop a decomposition scheme based on individual origins. For each origin, a spanning and acyclic sub-network is constructed and maintained throughout the iterations. Then a decomposed sub-problem is derived on each sub-network and solved using an efficient solution approach developed in the mathematical programming community. Numerical examples provided in the paper show that the proposed model and algorithm can solve efficiently medium size asymmetric traffic assignment problems and have the potential to be applied for large scale problems.

1. Introduction and Motivation

Following Wardrop's seminal work (1), numerous models and solution algorithms have been developed for the static traffic assignment problem, centered with the user equilibrium (2, 3). Majority of the solution algorithms have been focusing on the separable user equilibrium problem, based on the nonlinear programming (NLP) formulation by Beckmann et al. (4). Here, "separable" means there is no link interaction in terms of computing link travel times; in other words, the travel time of a link only depends on the traffic flow of the link itself. Although a well-defined NLP, Beckmann's formulation has a very special structure that the objective is defined on the aggregated variables (total link flows) and the constraints are defined on disaggregated variables (path flows in this case). Therefore, standard NLP solution techniques are not feasible for solving Beckmann's formulation directly, especially for large scale problems, since the dimension of the resulting NLP could be too large for any standard NLP solver to handle effectively.

In order to efficiently solve Beckmann's formulation, certain decomposition (or column generation) scheme has to be applied. In the literature, three major decomposition schemes have been developed, based on different aggregation levels, namely, the link-based, path-based, and origin-based. The Frank-Wolfe (FW) algorithm has a long history in the traffic assignment literature to solving Beckmann's formulation (5). FW is link-based and requires the minimum computer memory when implemented. However, the searching direction generated by FW tends to be perpendicular to the gradient of the objective function, implying the convergence rate of FW becomes very slow after certain number of iterations. Although many refinements of FW were proposed in the literature (6-8), it can only be used for producing approximate solutions. The gradient projection (GP) and disaggregated simplicial decomposition (DSD) are two path-based approaches. GP has different versions and the widely used one is due to Bertsekas (9) and Bertsekas and Gallager (10). In the transportation field, GP was firstly adopted by Jayakrishnan et al. (11) and later extensively studied and tested by Chen et al. (12). The DSD approach was proposed by Larsson and Patriksson (13). Both GP and DSD showed significant improvements in terms of solution accuracy and convergence efficiency compared with FW; however, they also need much more computer memory to store path-flows. The origin-based algorithm (OBA) for symmetric user equilibrium (14) reformulates Beckmann's model using origin-based disaggregated variables, thus having an aggregation level between the link-based and path-based algorithms. A distinctive feature of OBA is that it solves the traffic assignment on origin-based restricting sub-networks. Due to careful designs, OBA guarantees each sub-network is spanning and acyclic. Two advantages then follow. Firstly, the sub-network is generally smaller than the original network, implying it is more efficient to assign traffic flows on sub-networks. Further, since each sub-network is acyclic, no cyclic flow could be generated and network flow algorithms, e.g. finding the shortest path, can be much more efficiently performed. Therefore, OBA converges very quickly. Secondly, the memory requirement of OBA is much less than path-based algorithms because the aggregation level of OBA is much higher. Consequently, OBA can be used for solve large size traffic assignment problems (15) that path-based algorithms may not be able to handle.

Current implementation of OBA, however, targets on Beckmann's formulation only. Hence, it can not account for traffic interactions among different links. This is not as realistic as

the asymmetric (or non-separable) case where the travel time of a given link is dependent on both its own flow and those on the adjacent links. The asymmetric traffic assignment, particularly the asymmetric user equilibrium (AUE), has already been extensively studied. Due to the asymmetry feature of the problem, AUE can not be formulated (at least directly) as an NLP; rather, a variational inequality (VI) or nonlinear complementarity problem (NCP) formulation has to be adopted (16-20). In particular, all NCP models proposed so far for AUE are path based. To solve AUE models, a number of algorithms have been developed, including the diagonalization method (21), the decomposition methods (22, 23), and the simplicial decomposition methods (24). The diagonalization approach relaxes link interactions at each iteration of the algorithm and then the relaxed problem can be cast as a standard NLP. Dafermos (21) established conditions under which the iterative algorithm can converge globally. However, these conditions generally require strong monotonicity which may not be satisfied by traffic assignment problems. Aashtinai (22) and Pang (23) proposed decomposition schemes for solving asymmetric user equilibrium. In particular, Pang (23) provided conditions under which the schemes can converge locally or globally. The condition by Pang for local convergence only requires strict monotonicity which is weaker than those by Dafermos. Since the defining set of AUE is linear, its solution can be represented as a linear combination of extreme points of the defining set (a polyhedron). The simplicial decomposition thus aims to find these extreme points. Lawphongpanich and Hearn (24) established convergence conditions for simplicial decomposition and tested on small size problems. For detailed reviews of AUE models and algorithms, we refer to Patriksson (2).

All of the above AUE models and algorithms, nevertheless, were not fully tested for solving even medium size AUE problems. One reason is, as reported in Aashtinai (22), that no efficient solution algorithm for solving a general VI or NCP was available at the time when these algorithms were proposed. Recently, in the mathematical programming community, Ferris et al. (25) developed the path search algorithm which is considered as a break-through for solving large scale NCPs (26). The algorithm is globally convergent with a quadratic convergence rate. Therefore, how to derive a model for AUE so as to properly integrate the path search method to develop more efficient solution algorithms for AUE is an interesting research topic.

In this paper, we propose a complementarity model and a decomposition scheme for efficiently solving medium to large size AUEs. Firstly, we present a link-node based NCP formulation for AUE with a rigorous proof of the equivalence of the model and the user equilibrium condition. The solution existence and uniqueness conditions of the model are then established. To solve the model, we develop a decomposition scheme based on individual origins with a decomposed NCP solved for each origin. The path search algorithm is applied in the paper to solve the decomposed NCP. In order to further improve the efficiency, we construct and maintain a sub-network for each origin, similarly as the restricting sub-network in OBA. The sub-network is guaranteed to be spanning and acyclic so that the decomposed NCP can be solved very easily. Numerical examples conducted in the paper demonstrate that the proposed algorithm can solve efficiently medium scale AUE problems with high accuracy and has the potential to be applied for large scale problems.

This paper is organized as follows. The link-node based NCP formulation for AUE is presented in Section 2. This section also includes the equivalence proof of the model and the user

equilibrium condition, as well as establishing the solution existence and uniqueness conditions. In Section 3, the solution algorithm for the proposed model is discussed. It starts with a decomposition scheme based on individual origins. Then issues on how to construct and maintain sub-networks are addressed. Section 4 provides numerical examples demonstrating the effectiveness of the proposed algorithm. Finally, concluding remarks and future research directions are given in section 5.

2. Link-Node NCP Formulation for AUE

2.1 Link-Node NCP Formulation

As depicted in FIGURE 1, the user equilibrium route choice condition, whether symmetric or not, can be stated in a link-node fashion as follows.

If, from any origin to any decision node, the paths that are being used have equal and minimal travel times, then the traffic flow in the network is in a user equilibrium state.

Here a decision node with respect to an origin represents any node in the network except the origin itself. Mathematically, the above condition can be expressed as follows

$$\begin{cases} \pi_i^r + t_{ij}(\sum_{r \in R} u^r) - \pi_j^r > 0 \Rightarrow u_{ij}^r = 0, \forall r \in R, \forall (i, j) \in A, \\ \pi_i^r + t_{ij}(\sum_{r \in R} u^r) - \pi_j^r = 0 \Rightarrow u_{ij}^r \geq 0, \forall r \in R, \forall (i, j) \in A, \end{cases} \quad (1)$$

where R and A denote the set of origin nodes and all links, respectively, u_{ij}^r the disaggregated link flow on link (i, j) with respect to origin $r \in R$, $u^r = (u_{ij}^r)_{(i, j) \in A} \in \mathfrak{R}^{|A|}$, π_j^r the minimum travel time from origin r to a node j ($j \neq r$), and $t_{ij}(\sum_{r \in R} u^r)$ the link travel time for link (i, j) which is a function of the aggregated link flow $x = \sum_{r \in R} u^r$. It is easy to see that (1) is equivalent to the following complementarity condition:

$$0 \leq [\pi_i^r + t_{ij}(\sum_{r \in R} u^r) - \pi_j^r] \perp u_{ij}^r \geq 0, \forall r \in R, \forall (i, j) \in A. \quad (2)$$

Together with the flow conservation constraints at each node $j \in N$, we have the following formulation for AUE:

$$\begin{cases} 0 \leq [\pi_i^r + t_{ij}(\sum_{r \in R} u^r) - \pi_j^r] \perp u_{ij}^r \geq 0, \forall r \in R, \forall (i, j) \in A, \end{cases} \quad (3a)$$

$$\begin{cases} \sum_{i:(i,j) \in A} u_{ij}^r - \sum_{l:(j,l) \in A} u_{jl}^r - d_j^r = 0, \forall r \in R, \forall j \in N, j \neq r, \end{cases} \quad (3b)$$

$$\begin{cases} \pi_j^r \geq 0, \forall r \in R, \forall j \in N, j \neq r, \end{cases} \quad (3c)$$

where N denotes the set of nodes in the network and d_j^r the travel demand from origin r to node $j \in N$, $j \neq r$. Note that here we assume a node will never send trips to itself. So we set $d_r^r = 0$ and $\pi_r^r = 0$. It turns out that, as stated in Theorem 1, model (3) has the following NCP equivalence:

$$\begin{cases} 0 \leq [\sum_{i:(i,j) \in A} u_{ij}^r - \sum_{l:(j,l) \in A} u_{jl}^r - d_j^r] \perp \pi_j^r \geq 0, \forall r \in R, \forall j \in N, j \neq r, & (4a) \\ 0 \leq [\pi_i^r + t_{ij}(\sum_{r \in R} u^r) - \pi_j^r] \perp u_{ij}^r \geq 0, \forall r \in R, \forall (i,j) \in A, & (4b) \end{cases}$$

Theorem 1 If the link travel time function $t_{ij} = t_{ij}(\sum_{r \in R} u^r)$, $\forall (i,j) \in A$ is strictly positive for any $x = \sum_{r \in R} u^r \geq 0$ and the demand $d_j^r \geq 0, \forall r \in R, j \in N, j \neq r$, then model (3) is equivalent to the NCP model (4).

Proof. Denote $u^r = (u_{ij}^r)_{(i,j) \in A}$, $\pi^r = (\pi_j^r)_{j \in N, j \neq r}$, $u = (u^r)_{r \in R}$, $\pi = (\pi^r)_{r \in R}$. We need to prove that if (u, π) solves NCP (4) then it also solves (3) and vice versa.

a). It is obvious that any solution to (3) must also solve NCP (4).

b). Assume (u, π) solves NCP (4), we next show that it also solves (3). In order to do this, we use contradiction and suppose (u, π) is not a solution of (3). Then we must have at least one origin $r \in R$ and one node $j \in N, j \neq r$ such that $\sum_{i:(i,j) \in A} u_{ij}^r - \sum_{l:(j,l) \in A} u_{jl}^r - d_j^r > 0$. This will have two implications. Firstly, we must have $\pi_j^r = 0$ from (4a). Secondly, we have $\sum_{i:(i,j) \in A} u_{ij}^r > \sum_{l:(j,l) \in A} u_{jl}^r + d_j^r \geq 0$ since both u_{jl}^r and d_j^r are nonnegative. This in turn means that we must have at least one link going into node j , denoted as link (i,j) , such that $u_{ij}^r > 0$. Then from (4b), we can conclude that $\pi_i^r + t_{ij}(\sum_{r \in R} u^r) - \pi_j^r = 0$. Since we already proved $\pi_j^r = 0$, we have $\pi_i^r + t_{ij}(\sum_{r \in R} u^r) = 0$ implying $\pi_i^r = 0$ and $t_{ij}(\sum_{r \in R} u^r) = 0$ which contradicts our assumption that $t_{ij} = t_{ij}(\sum_{r \in R} u^r)$ is strictly positive.

From a) and b), we proved this theorem. □

Model (4) can be rewritten in a matrix form as:

$$\begin{cases} 0 \leq [\Lambda_r u^r - d^r] \perp \pi^r \geq 0, \forall r \in R, & (5a) \\ 0 \leq [-\Lambda_r^T \pi^r + t(\sum_{r \in R} u^r)] \perp u^r \geq 0, \forall r \in R, & (5b) \end{cases}$$

where $d^r = (d_j^r)_{j \in N, j \neq r} \in \mathfrak{R}^{|N|-1}$, $\pi^r = (\pi_j^r)_{j \in N, j \neq r} \in \mathfrak{R}^{|N|-1}$, and t is the link travel time vector which is a function of the total link flow vector $x = \sum_{r \in R} u^r$. Further denote $\Lambda \in \mathfrak{R}^{|N| \times |A|}$ as the link-node incidence matrix, then $\Lambda_r \in \mathfrak{R}^{(|N|-1) \times |A|}$ represents Λ with the row corresponding to origin r removed which essentially guarantees Λ_r a full row rank matrix.

2.2 Model Properties

In order to further study the mathematical properties of the NCP model (4), we first impose the following assumption on the link travel time function t .

Assumption 1 The link travel time t has the following properties:

- a) t is a smooth function of the total link flow $x = \sum_{r \in R} u^r$, i.e., $t = t(\sum_{r \in R} u^r)$.
- b) t is strictly monotone in terms of x , i.e., $(x_1 - x_2)^T [t(x_1) - t(x_2)] > 0$, $\forall x_1 \neq x_2$.
- c) $t(x) > 0$, $\forall x \geq 0$.
- d) $t(x) < \infty$ if $x < \infty$.
- e) t is a coercive function in terms of $u = (u^r)_{r \in R}$, i.e., $\lim_{\|u\| \rightarrow \infty} \frac{u^T t(u)}{\|u\|} = \infty$.

Note that under Assumption 1, the conditions needed for establishing Theorem 1 are satisfied. With these assumptions in place, we can obtain some nice properties regarding t which are listed in Lemma 1.

Lemma 1 The link travel time function has the following properties.

- a) The partial derivative of the link travel time to the disaggregated link flow variable corresponding to any origin is the same for a given total link flow vector x . That is,

$$\nabla_{u^r} t = \nabla_{u^s} t, \forall r, s \in R. \quad (6)$$

- b) For a given origin $r \in R$, if the disaggregated link flows with respect to all other origins are fixed, then the link travel time is a strictly monotone function in terms of u^r , i.e., $\nabla_{u^r} t, \forall r \in R$ is positive definite.

Proof. These two results can be straightforwardly derived from Assumption 1. □

We now are ready to prove NCP model (5) has at least one solution.

Theorem 2 The NCP model (5) has at least one solution if Assumption 1 holds and the AUE problem is feasible.

Proof. Due to the equivalence between (5) and (3), we only need to prove model (3) has at least one solution. For this purpose, rewrite (3) in a matrix form as:

$$\begin{cases} 0 \leq [-\Lambda_r^T \pi^r + t(\sum_{r \in R} u^r)] \perp u^r \geq 0, \forall r \in R, & \text{(i)} \\ \Lambda_r u^r - d^r = 0, \forall r \in R, & \text{(ii)} \\ \pi^r \geq 0, \forall r \in R, & \text{(iii)} \end{cases}$$

Denote a mapping $F: \mathfrak{R}^{|A||R|} \rightarrow \mathfrak{R}^{|A||R|}$ with $F = [t(u)^T \ \dots \ t(u)^T]_{1 \times |R|}^T$ and a set $K = \{u = (u^r)_{r \in R} \in \mathfrak{R}^{|A||R|} \mid \Lambda_r u^r - d^r = 0, u^r \geq 0, \forall r \in R\}$. Then $VI(K, F)$ is a linearly constrained VI since set K is a polyhedron. Obviously, $VI(K, F)$ has a nonempty and compact solution set according to Proposition 2.2.7 of Facchinei and Pang (26), provided Assumption 1 (e) is satisfied. On the other hand, according to the relationship between linear constrained VI and mixed complementarity problem (MiCP) described in Chapter 1 of Facchinei and Pang (26), the solution of $VI(K, F)$, denoted as \bar{u} , must satisfy the following MiCP for some $\bar{\pi} = (\bar{\pi}^r)_{r \in R}$:

$$\begin{cases} 0 \leq [-\Lambda_r^T \bar{\pi}^r + t(\sum_{r \in R} \bar{u}^r)] \perp \bar{u}^r \geq 0, \forall r \in R, & \text{(iv)} \\ \Lambda_r \bar{u}^r - d^r = 0, \forall r \in R, & \text{(v)} \end{cases}$$

where $\bar{\pi}^r \in \mathfrak{R}^{|N|-1}$, $\forall r \in R$ is the multiplier of the constraint $\Lambda_r u^r - d^r = 0$ in the definition of set K . Originally, $\bar{\pi}^r, \forall r \in R$ can be arbitrary; however, we next show that under (iv) and (v), we can always find $\hat{\pi} \geq 0$ such that $(\bar{u}, \hat{\pi})$ also satisfy (iv) and (v) where $\hat{\pi} = (\hat{\pi}^r)_{r \in R}$. To see this, we focus on one origin $r \in R$ and one node $j \in N, j \neq r$. Denote $f_j = \sum_{i(i,j) \in A} u_{ij}^r$ the total flow to node j . Apparently, $f_j \geq 0$ since all u_{ij}^r 's are nonnegative. Then, there are two cases regarding the value of f_j .

a). If $f_j > 0$, then there must exist a path from r to j such that this path will carry positive flows since we assume the problem is feasible. Denote this path as $r \rightarrow j_1 \rightarrow j_2 \rightarrow \dots \rightarrow j_n \rightarrow j$. We can obtain from equation (iv) that

$$\begin{cases} \bar{\pi}_{j_1}^r = \bar{\pi}_r^r + t_{rj_1} = 0 + t_{rj_1} \\ \vdots \\ \bar{\pi}_{j_n}^r = \bar{\pi}_{j_{n-1}}^r + t_{j_{n-1}j_n} \\ \bar{\pi}_j^r = \bar{\pi}_{j_n}^r + t_{j_nj} \end{cases}$$

Summing all these equations together, we can obtain $\bar{\pi}_j^r = t_{rj_1} + \sum_{k=1}^{n-1} t_{j_k j_{k+1}} + t_{j_n j} > 0$ since every individual link travel time is strictly positive according to Assumption 1 (c). In this case, we can set $\hat{\pi}_j^r = \bar{\pi}_j^r > 0$.

b) If $f_j = 0$, then all $\bar{u}_{ij}^r = 0, \forall (i, j) \in A$ and $\bar{u}_{jl}^r = 0, \forall (j, l) \in A$, and further $d_j^r = 0$. In this case all (iv) and (v) involved with node j are satisfied trivially which implies $\bar{\pi}_j^r$ can take any arbitrary value. However, in this case, we can set $\hat{\pi}_j^r = 0$.

Therefore, from a) and b), we can always find $\hat{\pi}_j^r \geq 0, \forall j \in R, j \in N, j \neq r$ such that $(\bar{u}, \hat{\pi})$ solves (iv) and (v), provided $(\bar{u}, \bar{\pi})$ satisfies (iv) and (v). In other words, we proved that (i), (ii), and (iii) has at least one solution, i.e. $(\bar{u}, \hat{\pi})$. \square

Several observations can be easily obtained for the NCP model (5). Firstly, it is a well-defined NCP problem which has at least one solution. Secondly, it is defined on the disaggregated link flow variables, while the link travel time function can be treated as a function of the total link flow. Thirdly, since the link travel time is a function of all $u^r, \forall r \in R$, the model (5) can represent AUE for which the link interactions need to be considered when computing the link travel time function.

The Jacobian matrix of NCP (5) is as follows.

$$M = \begin{bmatrix} \pi^1 & \dots & \pi^{|R|} & u^1 & \dots & u^{|R|} \\ 0 & \dots & 0 & \Lambda_1 & 0 & 0 \\ \vdots & \ddots & \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & 0 & 0 & \Lambda_{|R|} \\ -\Lambda_1^T & & & \nabla_{u^1} t & \dots & \nabla_{u^{|R|}} t \\ & \ddots & & & & \\ & & -\Lambda_{|R|}^T & \nabla_{u^1} t & & \nabla_{u^{|R|}} t \end{bmatrix} \begin{matrix} \pi^1 \\ \dots \\ \pi^{|R|} \\ u^1 \\ \dots \\ u^{|R|} \end{matrix}, \quad (7)$$

According to part a) of Lemma 1, we can denote $\nabla_{u^r} t = Q, \forall r \in R$ and the Jacobian can be rewritten as (8).

$$M = \begin{bmatrix} \pi^1 & \dots & \pi^{|R|} & u^1 & \dots & u^{|R|} \\ 0 & \dots & 0 & \Lambda_1 & 0 & 0 \\ \vdots & \ddots & \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & 0 & 0 & \Lambda_{|R|} \\ -\Lambda_1^T & & & Q & \dots & Q \\ & \ddots & & \vdots & \ddots & \vdots \\ & & -\Lambda_{|R|}^T & Q & \dots & Q \end{bmatrix} \begin{matrix} \pi^1 \\ \dots \\ \pi^{|R|} \\ u^1 \\ \dots \\ u^{|R|} \end{matrix}, \quad (8)$$

where Q is strictly monotone as stated in b) of Lemma 1.

Theorem 3 The Jacobian matrix M in (8) is positive semi-definite.

Proof. Clearly, M can be compactly written as:

$$M = \begin{bmatrix} 0 & B \\ -B^T & A \end{bmatrix}, \quad (9)$$

where $B = \begin{bmatrix} \Lambda_1 & & \\ & \ddots & \\ & & \Lambda_{|R|} \end{bmatrix}$ is full row rank since each $\Lambda_r, \forall r \in R$ is full row rank and

$A = \begin{bmatrix} Q & \cdots & Q \\ \vdots & \ddots & \vdots \\ Q & \cdots & Q \end{bmatrix}$ is positive semi-definite (see Lemma A4 in [27]). Then $\forall x, y$ with proper

dimensions, $[x^T \ y^T]M \begin{bmatrix} x \\ y \end{bmatrix} = [-y^T B^T \ x^T B + y^T A] \begin{bmatrix} x \\ y \end{bmatrix} = y^T A y \geq 0$ since A is positive semi-definite.

This implies M is a positive semi-definite matrix. \square

Note that M is not positive definite which implies that multiple solutions may exist for model (5). However, as stated in Theorem 4, the aggregated link flow must be unique.

Theorem 4. Denote $x = \sum_{r \in R} u^r$ the vector of total link flows. Assume part b) of Assumption 1 holds. Then NCP (5) has a unique solution in terms of the total link flow vector x .

Proof. Suppose (u_1, π_1) and (u_2, π_2) are any two solutions to NCP (5), we need to prove that

$\sum_{r \in R} u_1^r = \sum_{r \in R} u_2^r$. To see this, we note from (5b) that

$$\sum_{r \in R} (u_1^r)^Y [-\Lambda_r^T \pi_1^r + t(\sum_{r \in R} u_1^r)] = 0, \quad (10-a)$$

$$\sum_{r \in R} (u_2^r)^Y [-\Lambda_r^T \pi_1^r + t(\sum_{r \in R} u_1^r)] \geq 0, \quad (10-b)$$

$$\sum_{r \in R} (u_2^r)^Y [-\Lambda_r^T \pi_2^r + t(\sum_{r \in R} u_2^r)] = 0, \quad (10-c)$$

$$\sum_{r \in R} (u_1^r)^Y [-\Lambda_r^T \pi_2^r + t(\sum_{r \in R} u_2^r)] \geq 0. \quad (10-d)$$

We obtain by subtracting (10-b) from (10-a) that

$$\sum_{r \in R} (u_1^r - u_2^r)^T [-\Lambda_r^T \pi_1^r + t(\sum_{r \in R} u_1^r)] \leq 0. \quad (10-e)$$

Similarly, by subtracting (10-c) from (10-d), we get

$$\sum_{r \in R} (u_1^r - u_2^r)^T [-\Lambda_r^T \pi_2^r + t(\sum_{r \in R} u_2^r)] \geq 0. \quad (10-f)$$

We then obtain from (10-e)-(10-f) that

$$-\sum_{r \in R} [(u_1^r - u_2^r)^T \Lambda_r^T (\pi_1^r - \pi_2^r)] + \sum_{r \in R} [(u_1^r - u_2^r)^T (t(\sum_{r \in R} u_1^r) - t(\sum_{r \in R} u_2^r))] \leq 0. \quad (10-g)$$

Using the same derivation, we can obtain from (5a) that

$$\sum_{r \in R} (\pi_1^r)^T [\Lambda_r u_1^r - d^r] = 0, \quad (10-h)$$

$$\sum_{r \in R} (\pi_2^r)^T [\Lambda_r u_1^r - d^r] \geq 0, \quad (10-i)$$

$$\sum_{r \in R} (\pi_1^r)^T [\Lambda_r u_2^r - d^r] \geq 0, \quad (10-j)$$

$$\sum_{r \in R} (\pi_2^r)^T [\Lambda_r u_2^r - d^r] = 0. \quad (10-k)$$

We then obtain

$$\sum_{r \in R} [(\pi_1^r - \pi_2^r)^T \Lambda_r (u_1^r - u_2^r)] \leq 0. \quad (10-l)$$

Since $(u_1^r - u_2^r)^T \Lambda_r^T (\pi_1^r - \pi_2^r) = (\pi_1^r - \pi_2^r)^T \Lambda_r (u_1^r - u_2^r), \forall r \in R$, we can obtain by adding (10-g) and (10-l) that

$$\sum_{r \in R} [(u_1^r - u_2^r)^T (t(\sum_{r \in R} u_1^r) - t(\sum_{r \in R} u_2^r))] \leq 0. \quad (10-m)$$

Denote $x_1 = \sum_{r \in R} u_1^r$ and $x_2 = \sum_{r \in R} u_2^r$. Inequality (10-m) implies $(x_1 - x_2)^T [t(x_1) - t(x_2)] \leq 0$. Due to the strict monotonicity assumption of link travel time vector t with respect to the total link flow, we conclude that $x_1 = x_2$. \square

3. Solution Algorithm for AUE

3.1 A Decomposition Scheme

Although a well-defined NCP model, (5) is hard to solve directly for large scale multiple origin problems for two reasons. Firstly, the dimension of the problem could be too large (5); and secondly, the problem is “less monotone” with more origins. Ban (27) tested such a direct solve with certain regularity scheme by carefully adding proximal perturbation to the Jacobian matrix M in (8). However, it is still only feasible for solving small to medium size AUE problems. Ban (27) further tested the Gauss-Seidel decomposition method proposed by Pang (23), enhanced by a synchronization scheme to compute an optimal step size for each origin, and showed some promising results.

In this paper, we propose a scheme to apply the decomposition method on origin-based sub-networks. The algorithm decomposes the single NCP (5) into multiple smaller size NCP problems, one for each origin, by temporarily fixing the disaggregated link flows for all other origins. In other words, for each origin r , a decomposed NCP (DNCP) is constructed as follows.

$$\begin{cases} 0 \leq [\Lambda_r u^r - d^r] \perp \pi^r \geq 0, \\ 0 \leq \{-\Lambda_r^T \pi^r + t[\sum_{r' \in R, r' \neq r} \hat{u}^{r'} + u^r]\} \perp u^r \geq 0, \end{cases} \quad (11)$$

where $\hat{u}^{r'}$ is the temporarily fixed disaggregated link flow vector with respect to origin $r' \neq r$. Therefore, At each iteration, instead of solving directly the originally large dimensional NCP problem with size $(|A| + |N|) \cdot |R|$, we try to solve $|R|$ smaller dimensional NCP problems with size $(|A| + |N|)$. For most of the times, the path search algorithm can solve DNCP (11) very efficiently. The decomposition scheme is outlined as follows.

Step 1 Initialization

for each origin $r \in R$

Solve DNCP (11) by fixing $u^{r'}, \forall r' \neq r, r' \in R$ to zero if $r' > r$ and $(\bar{u}^{r'})^0$

if $r' < r$. Denote the solution as $((\bar{u}^r)^0, (\bar{\pi}^r)^0)$.

Set the iteration counter $n=0$;

Step 2 Main Loop

2.1 Solve DNCPs

for each origin $r \in R$

Perform DNCP (11) by fixing $\hat{u}^{r'}, \forall r' \neq r, r' \in R$ to $(\bar{u}^{r'})^n$ if $r' > r$ and $(\bar{u}^{r'})^{DNCP}$ if $r' < r$. Denote the solution as $((\bar{u}^r)^{DNCP}, (\bar{\pi}^r)^{DNCP})$.

2.2 Convergence checking

If certain convergence criterion is met, go to Step 3; otherwise, go to Step 2.3.

2.3 Update and Move

Set $(\bar{u}^r)^{n+1} = (\bar{u}^r)^{DNCP}, \forall r \in R$ and $n=n+1$, go to Step 2.1.

Step 3 Find an optimal solution $((\bar{u}^r)^{DNCP}, (\bar{\pi}^r)^{DNCP})$.

The above serial decomposition method will try to utilize the latest disaggregated link flow information from DNCP solves of other origins (Step 1 and 2.1). In the literature, it is also called the Gauss-Seidel (GS) decomposition (28).

3.2 Constructing Sub-Networks

To further improve the efficiency, we construct and maintain a sub-network for each origin. Then the DNCP can be performed on the sub-network, rather than the original entire network. The sub-network is guaranteed to be spanning and acyclic. Advantages of maintaining sub-networks are a) the sub-network is generally much smaller and therefore DNCPs can be solved more quickly, b) because each sub-network is acyclic, no cyclic flow could present, which will speed up the convergence of the algorithm, and c) network flow algorithms, e.g. finding the shortest path, are much more efficient (most likely to be linear with respect to the number of nodes or links of the network) on an acyclic network than on a general cyclic network. However, in order to ensure the spanning and acyclic-ness, sub-networks must be updated carefully.

Initially, the minimum spanning tree rooted from each origin can be used to construct the sub-network. Then, after solving all DNCPs for one round, each sub-network needs to be updated. Apparently, all used links, i.e. those with positive disaggregated link flow from last DNCP solve for the subject origin, should be included in the updated sub-network. Denote such a network as A_u^r (read as “used sub-network for origin r ”). Then we have

$$A_u^r = \{(i, j) \in A \mid u_{ij}^r > 0\}, \quad (12)$$

where u_{ij}^r denotes the current disaggregated link flow for link (i, j) with respect to origin r . Obviously, A_u^r is acyclic since the solution from DNCP (11) is so. Nevertheless, A_u^r may not be spanning. This can be illustrated in FIGURE 2 in which node 1 is the only origin and 6 is the only destination. Suppose after the latest DNCP solve for origin 1, only five links carry positive flow, namely, like $(1,2)$, $(1,3)$, $(2,5)$, $(3,5)$, and $(5,6)$, as marked bold in the figure. Clearly, the sub-network constructed by these five links, i.e., A_u^r , is not spanning since node 4 is not included. In order to add all missing nodes to the updated sub-network, we can first find the shortest path from the origin to the missing node, e.g. from 1 to 4 in FIGURE 2. Suppose path $1 \rightarrow 2 \rightarrow 4$ is the one we obtained. Then in order to include node 4 in the sub-network and also ensure the acyclic-ness, we can add the sub-path of the shortest path from the last node that is in current sub-network to the node that is not yet included, i.e. link $(2, 4)$ as shown in the dash line in the figure. More formally, we denote the set of these links as A_m^r (read as “missing links for origin r ”) and we have

$$A_m^r = \{(i, j) \in q = k_1 \rightarrow k_2 \cdots k_w \rightarrow k \mid q \subset p(r, k), \forall k \notin A_u^r; k_1 \in A_u^r; k_l \notin A_u^r, \forall l = 2, \dots, w\}, \quad (13)$$

where $p(r, k)$ denotes the shortest path from node r to node k in the sub-network. Note that in (13), $i \in A_u^r$ for node i actually means $\exists j \in N$ such that $(i, j) \in A_u^r$ or $(j, i) \in A_u^r$.

The updated sub-network $A_u^r \cup A_m^r$ is obviously spanning since all missing nodes are included. It is also acyclic. The reason is that $A_u^r \cup A_m^r$ can be constructed as follows. We start with A_u^r which is acyclic and for any node k not included in A_u^r , we add the path q as defined in (13). Since q is a sub-path of the shortest path $p(r,k)$, it is acyclic. Also because only the starting node of q is included in A_u^r , the cycle formed by adding q , if any, must include all links in q . However, this is impossible since there is no link in $A_u^r \cup q$ whose starting node is k (the ending node of path q). In the same reasoning, adding the sub-paths for all missing nodes to A_u^r will create no cycle.

To allow improved solutions, as discussed in Bar-Gera (14), we compute the maximum cost from the origin to each node in $A_u^r \cup A_m^r$ and add every link whose ending node has a higher cost than its starting node. Note that adding these links still guarantees that the sub-network is acyclic as proved in Bar-Gera (Lemma 8 in [14]). To sum-up, the updated sub-network A^r can be defined as follows, which is spanning and acyclic:

$$A^r = A_u^r \cup A_m^r \cup \{(i, j) \in A \mid v_j^r > v_i^r\}, \quad (14)$$

where v_i^r is the maximum cost from origin r to node i ($i \neq r$).

3.3 The Solution Algorithm

Finally, the solution algorithm for solving AUE, denoted as DECOMP, is listed as follows.

Step 1 Initialization.

For each origin r , construct the minimum spanning tree and perform the all-or-nothing assignment. This will generate the initial sub-network for the origin, denoted as A^r , and disaggregated link flow vector as $(\bar{u}^r)^0$. Set iteration counter $n=0$.

Step 2 Main Loop.

2.1 Solve DNCPs on Sub-Networks

For each origin $r \in R$

Solve DNCP (11) on A^r by fixing $\hat{u}^{r'}, \forall r' \neq r, r' \in R$ to $(\bar{u}^{r'})^n$ if $r' > r$ and $(\bar{u}^{r'})^{DNCP}$ if $r' < r$. Denote the solution as $((\bar{u}^r)^{DNCP}, (\bar{\pi}^r)^{DNCP})$.

2.2 Convergence Test

If certain convergence criterion is met, go to Step 3; otherwise, go to Step 2.3.

2.3 Update Sub-Networks

Update total link flow and link travel time for each link based on results from Step 2.1.

For each origin $r \in R$

2.3.1 Construct A_u^r as defined in (12). Use $(\bar{u}^r)^{DNCP}$ as the disaggregated link flow.

2.3.2 In the current (not updated) sub-network A^r , find the shortest path from r to any node that is not yet included in A_u^r . Construct A_m^r as defined in (13).

2.3.3 Find the maximum cost from r to any node $i \neq r$ in $A_u^r \cup A_m^r$, i.e. v_i^r . Construct A^r as defined in (14).

End

2.4 Update and Move

Set $(\bar{u}^r)^{n+1} = (\bar{u}^r)^{DNCP}$, $\forall r \in R$ and $n=n+1$, go to Step 2.1.

Step 3 Find an optimal solution, i.e. $(\bar{u}^r)^{DNCP}, (\bar{\pi}^r)^{DNCP}$.

Note that during solving the DNCP for origin r in Step 2.1, the latest information for solving DNCPs for all $r' < r$ is applied – the reason why it is called GS decomposition. Since the shortest path search in Step 2.3.2 and computing the maximum cost in Step 2.3.3 are performed on acyclic sub-networks, they can be done very efficiently, particularly, in a linear time with respect to the number of nodes in the sub-network. Further, the path search algorithm can solve very effectively the DNCPs on sub-networks. Therefore, DECOMP is very efficient as we will see in the next section.

Obviously DECOMP can produce origin-based link flows which are more detailed than solutions by link-based algorithms. On the other hand, the used paths and path flows can also be constructed by an enumeration procedure in each sub-network (14). Since the sub-network is acyclic, this enumeration procedure can be performed relatively efficiently.

4. Numerical Examples

In this section, we test the proposed NCP model and DECOMP algorithm on the Sioux-Falls and several other medium size networks. The purpose is to evaluate the effectiveness of the model and algorithm. All experiments were conducted on a PC with 3.8GHz CPU and 1GB memory, using the Windows XP operating system. The codes were written in C++. The path search algorithm is used to solve DNCPs. Especially, the PATH solver developed by Dirkse and Ferris (29) is adopted since this paper does not focus on developing solution algorithms for NCPs. PATH solver is called as an external package in current implementation.

4.1 Link Travel Time Function

Since we are dealing with AUE, we adopt the following link travel time function:

$$t_{ij}(x) = t_{ij}^0 \{1 + \alpha \cdot [(\sum_{(k,j) \in A} \rho_{ij,kj} x_{kj} + \sum_{(j,l) \in A} \rho_{ij,jl} x_{jl}) / (2 \cdot C_{ij})]^\beta\}, \quad (15)$$

where $0 \leq \rho_{ij,kj} \leq 1$ (or $\rho_{ij,jl}$) denotes the “impact factor” of the flow on link (k, j) (or (j, l)) to the travel cost of link (i, j) . Apparently, we have $\rho_{ij,ij} = 1, \forall (i, j) \in A$ and further if $\rho_{ij,kj} = \rho_{ij,jl} = 0, \forall j \in N, (i, j) \in A, (k, j) \in A, (j, l) \in A, i \neq k$, equation (15) will reduce to the standard BPR function. In this paper, we set $\rho_{ij,kj} = \rho_{ij,jl} = 0.15, \forall j \in N, (i, j) \in A, (k, j) \in A, (j, l) \in A, i \neq k$. α, β are constants and we set $\alpha = 0.15, \beta = 4$ in our study. We should point out here that in the literature, there are a number of ways to construct the link travel time for AUE (30). Equation (15) is

probably the most general expression which considers the impacts of all adjacent links to a given particular link.

4.2 Test Networks

We test on three networks in this paper, namely, the Sioux-Falls network, Anaheim network, and Winnipeg network. The network and OD data for Sioux-Falls network can be found in (31). The Anaheim network contains 416 nodes and 914 links with 38 OD zones (see (11) for data of the network). The Winnipeg network contains 1052 nodes and 2836 nodes with 147 OD zones. The network and OD data for this network can be obtained from (31). In our study, since we set $\alpha = 0.15, \beta = 4$, we change the capacity of each link to 2200. Table 1 summaries key characteristics of the three networks.

4.3 Results Analysis

To check the convergence of the algorithm, a modified version of the relative gap (14) is used in this study, as shown below.

$$\text{RelGap} = \frac{\sum_{(i,j) \in A} t_{ij}(\bar{x}) \cdot (\bar{x}_{ij} - x_{ij}^{AON})}{\sum_{(i,j) \in A} t_{ij}(\bar{x}) \cdot \bar{x}_{ij}}. \quad (16)$$

In equation (16), \bar{x} denotes the solution and x_{ij}^{AON} the all-or-nothing (AON) flow by fixing the link cost as $t_{ij}(\bar{x})$ for each link (i,j) . Apparently, from (16), the relative gap will always be nonnegative and if it goes to zero, the AUE problem is solved. Lawphongpanich and Hearn (24) used a very similar gap function as (16) for checking the convergence of their simplicial decomposition algorithm.

4.3.1 Convergence Performance

We list the convergence of the algorithm (in terms of the relative gap) with respect to both the number of iterations and the CPU time in this section. Note that in our implementation, PATH is called as an external program and data transferring is done through file I/O which takes majority of the running time. Therefore, for fairness, we exclude the file I/O time for the results shown in this paper. Future implementation of the algorithm will integrate the PATH solver within the C++ codes.

FIGURE 3 depicts the convergence of the algorithm with respect to the number of iterations for the three test networks. We can easily observe that for each of them, the relative gap converges quickly with the number of iterations increasing. In particular, we can see that the number of iterations needed for converging to a reasonably accurate solution (less than $1.0e-10$) is relatively small. For the Sioux-Falls network which is very small and has been extensively tested, using 100 iterations, the relative gap is reduced from 0.1 to $3.7e-13$. For the two medium size networks, the iterations required for convergence are around 40 and 30, respectively, for Anaheim and Winnipeg. These numbers are roughly the same as the restriction updates in OBA as reported in Bar-Gera (14). However, after updating each restricting sub-network once, OBA

requires multiple iterations (20 or more) to balance flows among links (paths). In our proposed DECOMP algorithm; nevertheless, only one DNCP solve is needed. The reason is that each DNCP solve (i.e. model (11)) aims to balance the distribution of origin-based demands according to the UE condition, by assuming origin-based link flows of other origins are temporarily fixed. Since we set the convergence threshold for DNCP solve to be pretty accurate ($1.0e-7$ is used in this paper) and thus the flows are well balanced. Therefore, there is no need to solve the DNCP for multiple times. Furthermore, as discussed in Larsson and Patriksson (13), solving the DNCP sub-problem with high accuracy will tend to speed up the convergence of the algorithm.

FIGURE 3 also lists the convergence of the relative gap with respect to CPU times. The same trends can also be observed, i.e. the algorithm converges quickly vs. CPU times. Table 1 further summarizes the detailed convergence data for the three test examples. Normally, it takes DECOMP seconds to solve small size AUEs and minutes on medium size networks.

4.3.2 Equilibrium Solutions

The solution of DECOMP contains origin-based link flows which can provide more detailed information than link-based algorithms. Further, path flows for all used paths can also be constructed by performing a path-enumeration search in each sub-network. FIGURE 4 – 5 list the origin-based link flows for origin 1 and 17, respectively, for Sioux Falls. We can see that the sub-network for origin 1 is exactly a tree structure with only one path from node 1 to any other nodes. However, for origin 17, a relatively complicated network is formed. We can easily observe that there are five paths from origin 17 to destination 12 in FIGURE 5. For comparison purposes, FIGURE 6 depicts the origin-based link flow solution for origin 1 of Sioux Falls for the **symmetric** case. Note that our proposed DECOMP can solve both symmetric and asymmetric UE problems. Comparing FIGURE 6 and FIGURE 29 in (14), we can see that our approach produces almost the identical solution as that by OBA. Table 1 further lists more detailed information for origin-based solutions for the three test examples. For this table, we can observe that averagely each sub-network only contains a small portion of links in the original network (from 8% to 30% for the test examples). That is why the DNCPs can be very efficiently solved by the path search algorithm.

4.3.3 Memory Requirement

This paper mainly focuses on the efficiency of the origin-based algorithm and the implementation may not be the one with least memory usage. Table 1 demonstrates the memory usage for the three test examples. From this table, we can see that for medium size networks like Winnipeg, only 20 MB memory is used. Therefore, the memory usage for DECOMP is relatively small. Combined with its high efficiency in terms of convergence, the algorithm has the potential to be applied for large scale problems.

5. CONCLUSIONS

In this paper, we proposed a complementarity model and a decomposition scheme for solving asymmetric user equilibrium. We first presented a link-node based NCP formulation for AUE with a proof of the equivalence of the model and the user equilibrium condition. We proved that

the model has at least one solution and under mild conditions, the aggregated link flow is unique. Built on this new formulation, a decomposition scheme was developed based on individual origins to convert the single large dimensional NCP into multiple smaller size decomposed NCPs. DCNPs can be efficiently by the path search algorithm developed in the mathematical programming community. To further improve the efficiency, we constructed a sub-network for each origin and solved the DNCP on the sub-network. The sub-network was carefully constructed and updated such that it remains spanning and acyclic. The proposed algorithm thus combines the strengths of path search algorithm and the basic ideas of the origin-based algorithm for symmetric user equilibrium. Numerical examples showed that the proposed algorithm can solve medium size AUE with rapid convergence and high accuracy, and has the potential to solve large scale AUE problems.

For future study, we need to rigorously prove the convergence of the proposed algorithm. Pang (23) and Bar-Gera (14) provided starting points for such a topic. The authors are currently investigating conditions under which the proposed scheme converges and the results will be reported in subsequent papers. This paper tested the algorithm on selected medium size AUE problems. Given the potential of the proposed model and algorithm for solving large scale problems, extensive evaluations of the algorithm are still needed in the future study. Especially we will compare our algorithm with existing approaches for solving medium size AUE and further test it for solving large scale problems.

References

1. Wardrop, J.G. (1952) Some theoretical aspects of road traffic research. In *Proceedings of the Institution of Civil Engineers*, Part II, 1, 325-378.
2. Patriksson, M. (1994) *The Traffic Assignment Problem, Models and Methods*. VSP.
3. Boyce, D.E., Mahmassani, H.S., and Nagurney, A. (2005) A retrospective on Beckmann, McGuire and Winsten's Studies in the Economics of Transportation. *Papers in Regional Science*, in press.
4. Beckmann, M.J., McGuire, C.B., and Winsten, C.B. (1956) *Studies in the Economics of Transportation*. Yale University Press, New Haven, Connecticut.
5. LeBlanc, L.J., Morlok, E.K., and Pierskalla, W.P. (1975) An efficient approach to solving the road network equilibrium traffic assignment problem. *Transportation Research* 9, 309-318.
6. Florian, M., and Spiess, H. (1983) Transport networks in practice. *Proceedings of the Conference of the Operations Research Society of Italy*, Napoli, 29-52.
7. Fukushima, M. (1984) A modified Frank-Wolfe algorithm for solving the traffic assignment problem. *Transportation Research* 18B, 169-177.
8. LeBlanc, L.J., Helgason, R.V., and Boyce, D.E. (1985) Improved efficiency of the Frank-Wolfe algorithm for convex network programs. *Transportation Science* 19, 445-462.
9. Bertsekas, D. (1976) On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Trans. Auto. Control*, 21, 174-183.
10. Bertsekas, D.P., and Gallager, R.G. (1987). *Data Networks*. Englewood Cliffs, NJ: Prentice-Hall.
11. Jayakrishnan, R., Tsai, W.K., Prashker, J.N., and Rajadhyaksha, S. (1994). A faster path-based algorithm for traffic assignment. *Transportation Research Record*, 1443, 75-83.

12. Chen, A., Lee, D.H., and Jayakrishnan, R. (2002) Computational study of state-of-the-art path-based traffic assignment algorithms. *Mathematics and Computers in Simulation* 59, 509-518.
13. Larsson, T., and Patriksson, M. (1992) Simplicial decomposition with disaggregated representation for the traffic assignment problem. *Transportation Science*, 26, 4-17.
14. Bar-Gera, H. (1999) *Origin-based algorithm for transportation network modeling*. Ph.D thesis, University of Illinois at Chicago, Chicago, IL.
15. Boyce, D.E., Ralevic-Dekic, B., and Bar-Gera, H. (2004) Convergence of traffic assignment: How much is enough? *ASCE Journal of Transportation Engineering* 130(1), 49-55.
16. Smith, M.J. (1979). The Existence, Uniqueness and Stability of Traffic Equilibria. *Transportation Research* 13B, 295-304.
17. Dafermos, S.C. (1980) Traffic Equilibrium and Variational Inequality. *Transportation Science* 14, 42-54.
18. Aashtiani, H.Z., and Magnanti, T.L. (1981) Equilibria on a congested transportation network. *SIAM Journal on Algebraic and Discrete Methods* 2(3), 213-226.
19. Friesz, T.L., Tobin, R.L., and Smith, T.E., etc. (1983) A nonlinear complementarity formulation and solution procedure for the general derived demand network equilibrium problem. *Journal of Regional Science* 23, 337-359.
20. Nagurney, A. (1998) *Network Economics: A Variational Inequality Approach (2nd Edition)*. Kluwer Academic Publisher.
21. Dafermos, S.C. (1983) An iterative scheme for variational inequalities. *Mathematical Programming* 26, 40-47.
22. Aashtiani, H.Z. (1979) *The Multi-Modal Traffic Assignment Problem*. Ph.D. Thesis, MIT, Cambridge, MA.
23. Pang, J.S. (1985) Asymmetric variational inequality problems over product sets: applications and iterative methods. *Mathematical Programming* 31, 206-219.
24. Lawphongpanich, S., and Hearn, D. (1984) Simplicial decomposition of the asymmetric traffic assignment problem. *Transportation Research* 18B, 123-133.
25. Ferris, M. C., Kanzow, C., and Munson, T. S. (1999) Feasible descent algorithms for mixed complementarity problems. *Mathematical Programming* 86, 475-497.
26. Facchinei, F., and Pang, J.S. (2003) *Finite-Dimensional Variational Inequalities and Complementarity Problems (Vol. I and II)*. Springer-Verlag, New York.
27. Ban, X. (2005) *Quasi-Variational Inequality Formulations and Solution Approaches for Dynamic User Equilibria*, Ph.D. Thesis, University of Wisconsin-Madison, Madison, Wisconsin.
28. Bertsekas, D.P., and Tsitsiklis, J.N. (1997) *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, London.
29. Dirkse, S.P., and Ferris, M.C. (1995) The PATH solver: A non-monotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software* 5, 123-156.
30. Nagurney, A. (1984) Comparative test of multimodal traffic equilibrium methods, *Transportation Research* 18B, 469-485.
31. Transportation Network Test Problems, URL: <http://www.bgu.ac.il/~bargera/tntp/>.

List of Tables

Table 1 Data and Results of Test Examples

List of Figures

FIGURE 1 Illustration of UE Condition

FIGURE 2 Illustration of Updating the Sub-Network: to Guarantee Spanning

FIGURE 3 Convergence Performance for Test Examples

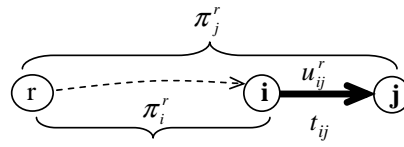
FIGURE 4 Sioux Falls Equilibrium Solution – Link Flows for Origin 1

FIGURE 5 Sioux Falls Equilibrium Solution – Link Flows for Origin 17

FIGURE 6 Sioux Falls Equilibrium Solution – Link Flows for Origin 1 (Symmetric Case)

Table 1 Data and Results of Test Examples

Network	Sioux Falls	Anaheim	Winnipeg
Zones (Origins)	24	38	147
Nodes	24	416	1052
Links	76	914	2836
OD Pairs	528	1406	4345
CPU Time	1 second	1 minutes	18 minutes
# of Iterations	100	40	30
Relative Gap	3.73E-13	3.10E-10	4.90E-11
Links with flow > capacity	58 (76%)	62 (6.8%)	152 (5.4%)
Origin-based used links	572	6952	34751
Average links in one sub-network	23.8 (31.3%)	182.9 (20.0%)	236.4 (8.3%)
Total # of routes	585	1596	4477
Average routes per OD pair	1.11	1.14	1.03
Maximum routes per OD pair	5	5	2
Average links per route	3.2	17.2	25.8
Memory Usage	0.9 MB	9.5 MB	20.4 MB

**FIGURE 1** Illustration of UE Condition

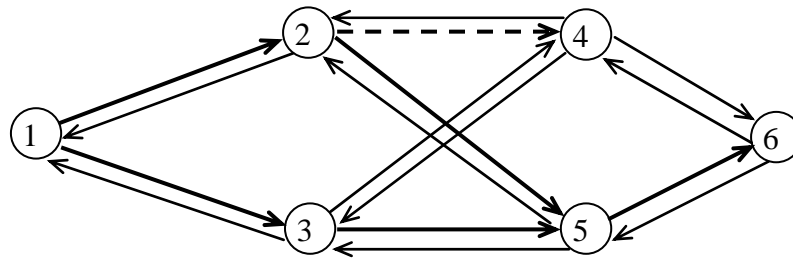


FIGURE 2 Illustration of Updating the Sub-Network: to Guarantee Spanning

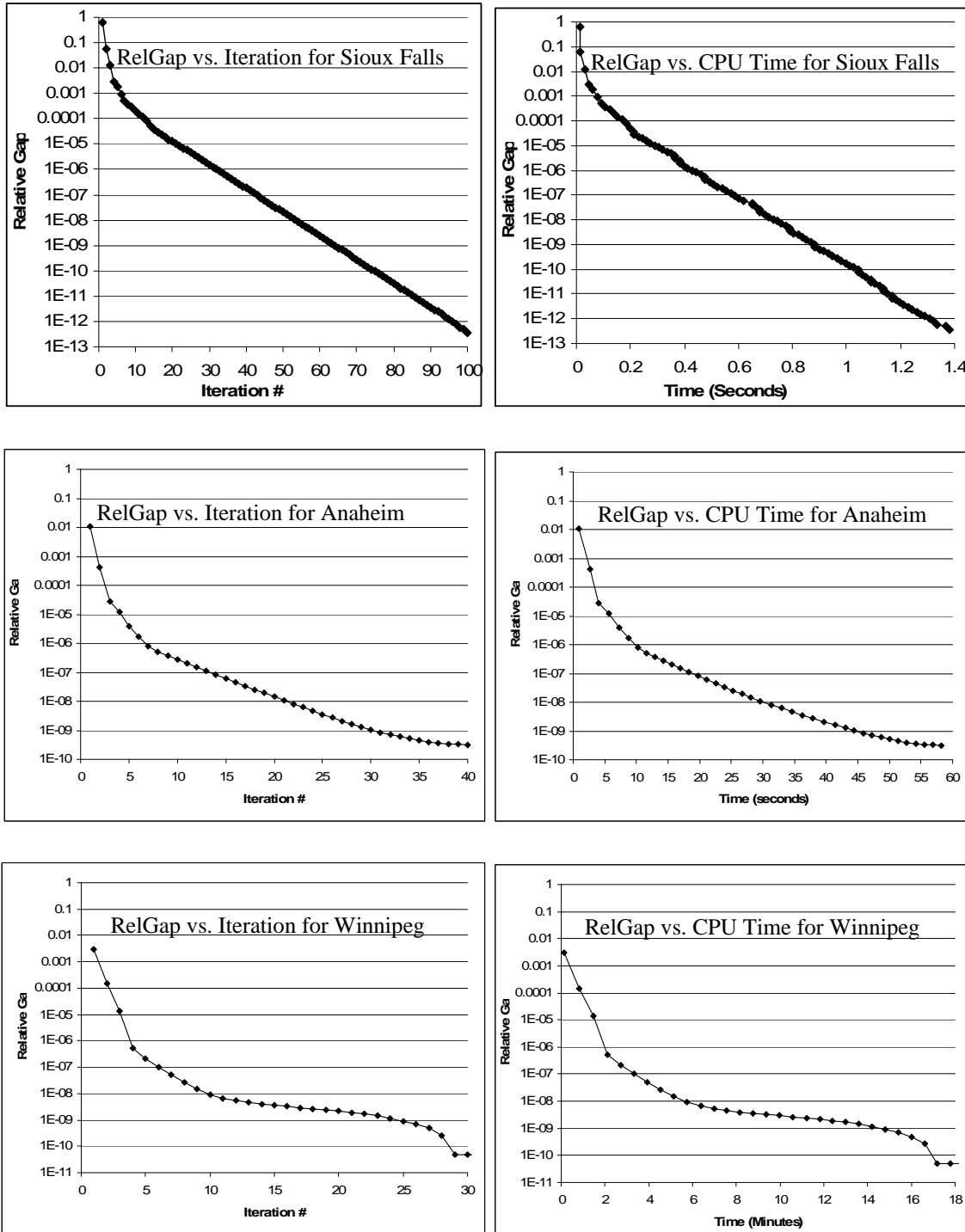


FIGURE 3 Convergence Performance of Test Examples

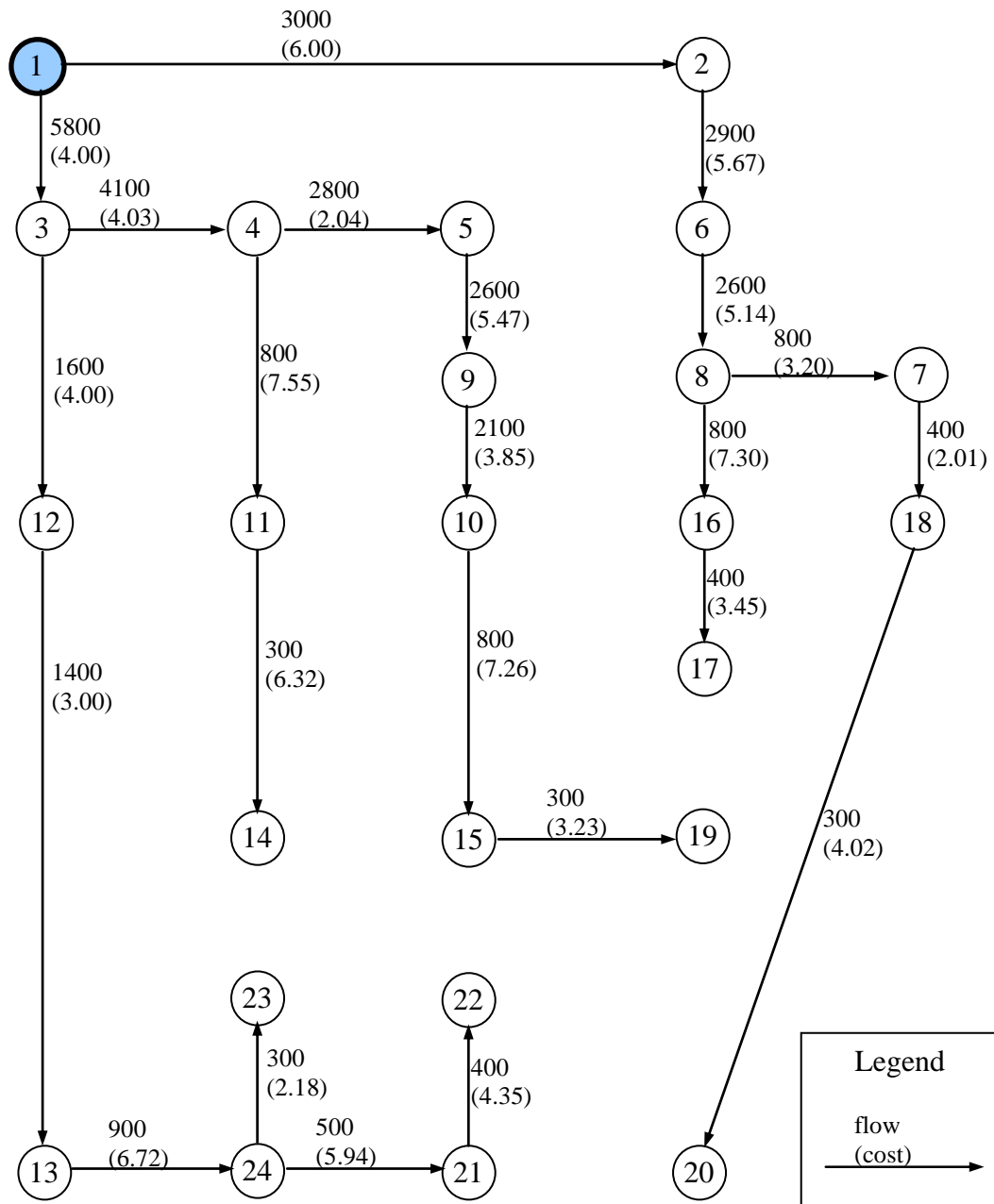


FIGURE 4 Sioux Falls Equilibrium Solution – Link Flows for Origin 1

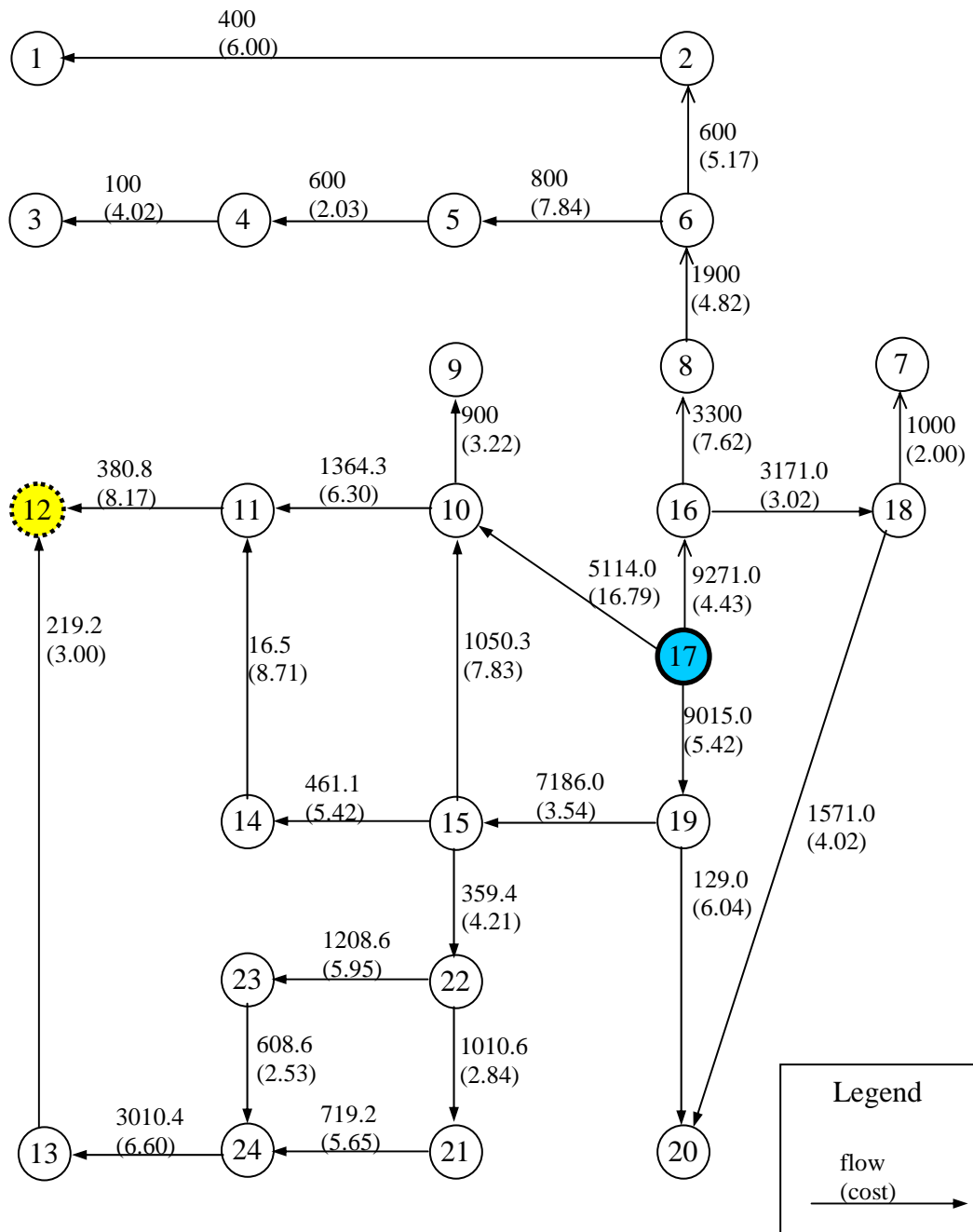


FIGURE 5 Sioux Falls Equilibrium Solution – Link Flows for Origin 17

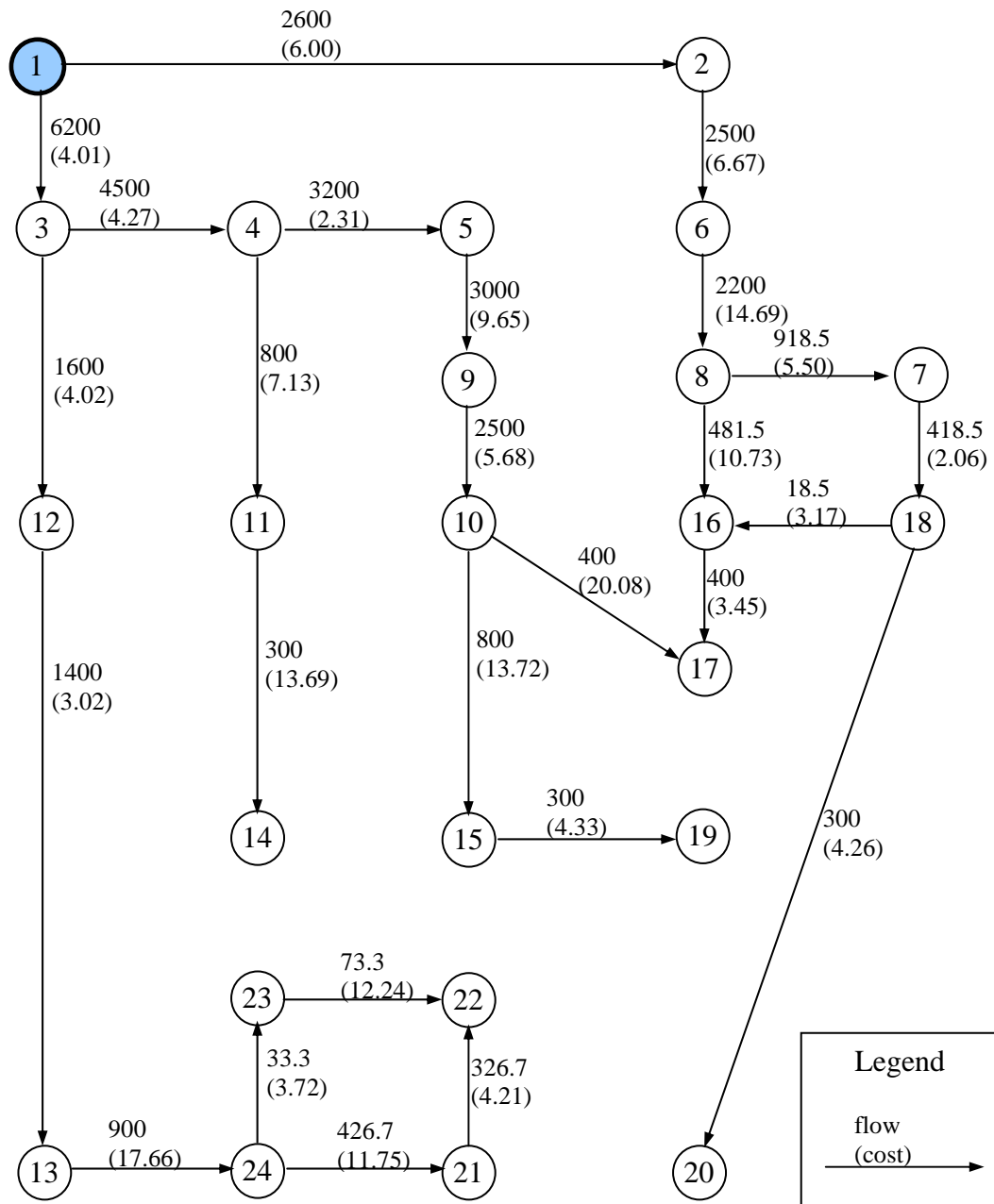


FIGURE 6 Sioux Falls Equilibrium Solution – Link Flows for Origin 1 (Symmetric Case)