Primal-Dual Heuristic for Path Flow Estimation in Medium to Large Networks

Shikai Tang  
Graduate Student Researcher  
Institute of Transportation Studies  
University of California at Davis  
Davis, CA 95616

H. Michael Zhang  
Professor of Civil and Environmental Engineering  
University of California at Davis  
Davis, CA 95616  
Email: hmzhang@ucdavis.edu  
And  
Distinguished Professor of Transportation Engineering  
School of Transportation Engineering  
Tongji University, Shanghai, China 201804

Submission Date: 08/01/2012

Number of words: 4962

Number of figures and tables: 7

Total: 5354+8*250 = 7354

Submitted for presentation at the 92th Transportation Research Board Annual Meeting, January 13-17, 201
Abstract

The Path Flow Estimator (PFE), an Origin-Destination (OD) demand estimation algorithm that relies on the computation of path flows, can be slow when applied to medium to large networks. We develop a primal-dual heuristic that can significantly improve the computational efficiency of the algorithm when it is applied to large networks. Numerical examples are provided to show the performance improvement of this primal-dual heuristic over the original PFE algorithm.

Key words: Path Flow Estimator (PFE), primal-dual heuristic, OD estimation for large networks
Introduction

The Origin-Destination (OD) trip table is an essential input to many transportation applications and numerous efforts have been devoted to develop models that estimate OD travel demand from traffic counts (see [1-3] for a review on modeling, algorithm development and application on this topic). The earlier contributions along this direction are credited to Willumsen [4] for introducing entropy maximization, Van Zuylen [5] for the concept of information minimization, and Cascetta [6] for casting the problem in a generalized least square framework. After decades of research, various OD demand estimators have been proposed. In general, these estimators could be classified into two categories, equilibrium-based or non equilibrium-based. If the network concerned has no or minor congestion, i.e., travel time is almost independent of link traffic flow, its demand pattern is assumed to be exogenously determined and obtained by means of proportional loading procedures, which can for example be a simple all-or-nothing (AON) assignment. We call this type of models proportional-assignment OD estimation (ODE) models. Many researchers, such as Willumsen [4], Van Zuylen [5], Cascetta [6], Bell [7-8], McNeil and Hendrickson [9] and Lo et al. [10] have proposed proportional-assignment ODE models. However, as congestion prevails in a network, the proportional-assignment assumption no longer holds. To take into account network congestion, equilibrium assignment models are incorporated into OD demand estimation. We call this type of models the equilibrium-assignment OD estimation (E/ODE) models. The first E/ODE model, due to Nguyen [11] [12], is to find an OD matrix that reproduces the observed travel cost when assigned onto the network in a user-equilibrium fashion. However, the solution of Nguyen’s estimation model may not be unique since the optimization problem is not strictly convex with respect to OD demands. That means there are multiple OD trip matrices that can produce the same link flows. Consequently, a great deal of work has been focused on introducing a secondary optimization problem that leads to a unique solution by minimizing the squared difference between the estimated OD table with a target/historical OD table (Turnquist & Gur [13], LeBlanc & Farhangian [14], Sheffi [15], Yang et al. [16], Yang [17]), maximizing the likelihood of the estimated OD table (Spiess [18], Maher [19]), or maximizing an entropy function that leads to the most likely OD table (Fisk & Boyce [20], Fisk [21]).

Because of the second requirement of E/ODE models, they are widely cast into bi-level form. Models of Yang et al. [16], Yang [17], and Maher et al. [22]’s are examples of this formulation. Although bi-level formulations could capture equilibrium traffic distribution more accurately, they are very challenging to solve and there is no guarantee that the solution found numerically is the global solution. In order to avoid those challenges, path flow estimators (PFE) based on single-level optimization, in which path flows instead of OD demands are treated as solution variables, are proposed. Sherali et al. [23] develop the first path flow estimator as a linear programming problem. The path decomposition of OD flows that reproduce the observed link flows is determined by a column generation procedure such that the resulting OD table is the closest to a target (historical) one, while link traffic flows are loaded to approximate user equilibrium assignment. Nie and Lee [24] suggest replacing the column generation procedure by
a K-shortest path ranking algorithm that determines those quasi-UE paths (columns) exogenously. Nie, Zhang and Recker [25] incorporate this decoupled structure into a generalized least squares (GLS) framework to develop a GLS path flow estimator. In this paper, we study another path flow estimator proposed by Bell [26]—the logit path flow estimator (LPFE), the core of which is a logit-based path choice model. As the result of its convex programming formulation, LPFE gives unique path flows and does not require traffic volumes to be measured on all network links (constraints of linear PFE [23-25]) but assuming that each unmeasured link has an explicit capacity. This simple mathematical structure of LPFE ensures a global convergence to a unique solution but has the drawback of enumeration of all possible paths between any OD pair, which makes the convergence slow when it is used in medium to large networks.

In this paper, we develop a primal-dual heuristic that can significantly speed up the convergence of LPFE and enable its application in medium to large networks with considerably less computational time. In the following sections, we present the original LPFE model and its solution algorithm. We describe our primal-dual heuristic to improve the original solution algorithm. We demonstrate the effectiveness of our primal-dual heuristic with some numerical results and summarize them in the conclusion.

The Mathematical Formulation of LPFE and its Solution Algorithm

As given in Bell [26] and Chootinan et al. [27], the LPFE model is formulated as follows (See Appendix for an explanation of the notations):

$$\min Z(f) = \frac{1}{\theta} \sum_{rs \in D} \sum_{k \in K^{rs}} f_{rk} \left( \log f_{rk} - 1 \right) + \sum_{a \in A} \int_0^{x_a} t_a(\omega) d\omega$$  \hspace{1cm} (1)

s.t.

$$x_a \leq (1 + \varepsilon_a) \cdot v_a \quad \forall a \in M; \quad \lambda_a^+$$  \hspace{1cm} (2)

$$(1 - \varepsilon_a) \cdot v_a \leq x_a \quad \forall a \in M; \quad \lambda_a^-$$  \hspace{1cm} (3)

$$x_a \leq C_a \quad \forall a \in U; \quad \mu_a$$  \hspace{1cm} (4)

$$x_a = \sum_{rs \in RS} \sum_{k \in K^{rs}} f_{rk}$$

$$\sum_{k \in K^{rs}} f_{rk} \leq D_{rs} \cdot (1 + \varepsilon_{rs}) \quad \forall rs \in D; \quad \phi_{rs}^+$$  \hspace{1cm} (5)

$$D_{rs} \cdot (1 - \varepsilon_{rs}) \leq \sum_{k \in K^{rs}} f_{rk} \quad \forall rs \in D; \quad \phi_{rs}^-$$  \hspace{1cm} (6)

Where $\theta$ is called the dispersion parameter which represents travelers’ sensitivity to path costs. If $\theta \to \infty$, the estimated path flows are close to user equilibrium. If $\theta \to 0$, the estimated path flows are close to entropy maximization.
In order to describe the solution algorithm of LPFE, we need to form the Lagrangian with dual variables $(\lambda^+, \lambda^-, \mu, \varphi^+, \varphi^-)$:

$$L(f, \lambda^+, \lambda^-, \mu, \varphi^+, \varphi^-) = \frac{1}{\theta} \sum_{rs \in D} \sum_{k \in K_{rs}} f^r_k (\log f^r_k - 1) + \sum_{a \in A} \int_0^{x_a} t_a(\omega) d\omega$$

$$+ \sum_{a \in M} \left[ \sum_{rs} \sum_k f^r_k \delta^r_{ka} - (1 + \epsilon_a) \cdot v_a \right] \cdot \lambda^+$$

$$+ \sum_{a \in M} \left[ (1 - \epsilon_a) \cdot v_a - \sum_{rs} f^r_k \delta^r_{ka} \right] \cdot \lambda^- + \sum_{a \in M} \left[ \sum_{rs} f^r_k \delta^r_{ka} - C_a \right] \cdot \mu_a$$

$$+ \sum_{k \in K_{rs}} f^r_k - D_{rs} \cdot (1 + \epsilon_{rs}) \cdot \varphi^+ + \left[ D_{rs} \cdot (1 - \epsilon_{rs}) - \sum_{k \in K_{rs}} f^r_k \right] \cdot \varphi^-$$

(7)

$$\lambda^+ \geq 0, \lambda^- \geq 0, \mu, \varphi^+, \varphi^- \in \mathbb{R}^+$$

The derivative of function $L$ with respect to $f^r_k$ is:

$$\frac{\partial L}{\partial f^r_k} = \frac{1}{\theta} \log f^r_k + \sum_{a \in A} t_a(x_a) \delta^r_{ka} + \sum_{a \in M} \sum_{rs} \sum_k \delta^r_{ka} \cdot \lambda^+ + \sum_{a \in M} \sum_{rs} \sum_k \delta^r_{ka} \cdot \lambda^-$$

$$+ \sum_{a \in M} \sum_{rs} \sum_k f^r_k \delta^r_{ka} \cdot \mu_a + \varphi^+ \cdot \varphi^-$$

$$\forall k \in K_{rs}, rs \in RS$$

(9)

The KKT optimality condition is then given by:

$$\frac{\partial L}{\partial f^r_k} = 0 \Rightarrow f^r_k = \exp \left( -\theta \sum_{a \in A} t_a(x_a) \delta^r_{ka} - \theta \sum_{a \in M} \sum_{rs} \sum_k \delta^r_{ka} \cdot \lambda^+ \right.$$  

$$+ \theta \sum_{a \in M} \sum_{rs} \sum_k \delta^r_{ka} \cdot \lambda^- - \theta \sum_{a \in M} \sum_{rs} \sum_k \delta^r_{ka} \cdot \mu_a - \theta \varphi^+ + \theta \varphi^- \right)$$

(10)

$$0 \leq \lambda^+ \perp \sum_{rs} f^r_k \delta^r_{ka} - (1 + \epsilon_a) \cdot v_a \leq 0$$

(11)

$$0 \leq \lambda^- \perp (1 - \epsilon_a) \cdot v_a - \sum_{rs} f^r_k \delta^r_{ka} \leq 0$$

(12)

$$0 \leq \mu_a \perp \sum_{rs} f^r_k \delta^r_{ka} - C_a \leq 0$$

(13)

$$0 \leq \varphi^+ \perp \sum_{k \in K_{rs}} f^r_k - D_{rs} \cdot (1 + \epsilon_{rs}) \leq 0$$

(14)

$$0 \leq \varphi^- \perp D_{rs} \cdot (1 - \epsilon_{rs}) - \sum_{k \in K_{rs}} f^r_k \leq 0$$

(15)
The original solution algorithm suggested by Bell [22], Chootinan et al. [25] and Chen et al. [28] utilizes the following iterative procedure to update path flows and the objective function value (See Figure 1 for flowchart of the original algorithm):

Step 1. **Find shortest paths for every target OD pair rs ∈ D based on current link flow cost.**

Step 2. **Giving dual variables (λ⁺, λ⁻, μ, ϕ⁺⁻, ϕ⁻⁻) and their initial values (0),**

- update path flows according to equation (10);
- Adjust dual variables’ values and recalculate the path flows, until constraints (2) – (6) and (8) are all satisfied or the objective function value could not be improved under current link flow cost;
- Update link costs according to current path flows and find alternative path flow patterns that reduce the objective function (1) by repeating Step 2 from the beginning with the updated link flow cost until minimum objective function value is reached under current path set.

Step 3. **If there are still some paths that have lower path cost compared with paths in current path set, insert those paths into current path set and go back to Step 2.**

Figure 1. Flowchart of the original algorithm (Chen et al. [28]) and the primal-dual heuristic
In above algorithm, Step 2 is most computationally intensive, in which the procedure of updating dual variables, path flows and link costs repeats multiple times. When combined with a line search procedure, this step becomes extremely slow especially applied to large networks. Hence, the key to improve the overall efficiency of the algorithm is to speed up Step 2 by improving its dual variables updating process. Chootinan et al. [24] notice this possibility of efficiency improvement and suggest a simple heuristic to update the dual variables more efficiently. That heuristic, however, does not take full advantage of the complementary relationship between link costs \((t(x))\) and dual variables \((\lambda^+, \lambda^-, \mu, \varphi^+, \varphi^-)\) in equation (10), i.e., increasing link costs to some extent would be equivalent to decreasing \(\lambda^+, \mu\) or increasing \(\lambda^-\) by a certain amount when calculating path flows according to (10). Instead of updating dual variables from previous results, the practice of re-initializing dual variables in Step 2 of the original algorithm takes default values \((0)\) for all dual variables in every loop. Hence, we suggest a new primal-dual heuristic, which adopts information obtained from previous iteration to update dual variables, to speed up the convergence of the whole algorithm. Through the proposed primal-dual heuristic, the algorithm would avoid redundant computation on dual variables to retain a sub-optimal path flow pattern (See definition 2) obtained by previous computation.

We use the following example as shown in Figure 2 (Red color stands for sub-optimal path flows) to explain this process in more detail:

**Definition 1.** If path flows \((f^{rs}_{1}, f^{rs}_{2}, \ldots, f^{rs}_{k}, rs \in D, k \in K^{rs})\) satisfy the following conditions:

1) (Feasible Path Flows) Satisfy the link and historical OD demand constraints (2)-(6).
2) Minimize the objective function with link costs fixed:

\[
\min \left\{ \sum_{s \in D} \sum_{k \in K^{rs}} f^{rs}_{k} (\log f^{rs}_{k} - 1) + \sum_{a \in A} c_{a} x_{a} \right\}
\]

Here, \(c_{a} (a \in A)\) stands for the fixed link cost after some feasible path flow loading.

Then, these path flows are called **Fixed Link Cost Optimal (Local Optimal) Path Flows.**

**Definition 2.** If path flows \((f^{rs}_{1}, f^{rs}_{2}, \ldots, f^{rs}_{k}, rs \in D, k \in K^{rs})\) satisfy the following conditions:

1) (Feasible Path Flows) Satisfy the link and historical OD demand constraints (2)-(6).
2) Within them, some subset of path flows \((f^{r's'}_{1}, f^{r's'}_{2}, \ldots, f^{r's'}_{k}, k' \in K^{r's'}, R^{r's'} \subseteq D^*, D^* \subseteq D)\) minimize the partial path flow objective function:

\[
\min \left\{ \sum_{r's' \in D^*} \sum_{k' \in K^{r's'}} f^{r's'}_{k'} (\log f^{r's'}_{k'} - 1) + \sum_{a \in A_{D^*}} \int_{0}^{x_{a}} t_{a}(\omega) d\omega \right\}
\]

Here, \(D^*\) could be any non empty subset of \(D\); \(A_{D^*}\) stands for links crossed by the path subset.

Then, this subset of path flows \((f^{r's'}_{1}, f^{r's'}_{2}, \ldots, f^{r's'}_{k}, k' \in K^{r's'}, R^{r's'} \subseteq K^{r's'}, r's' \in D^*, D^* \subseteq D)\) is called **Partial Optimal (Sub-Optimal) Path Flows.**
Default Dual Value: $\lambda_1^-, \lambda_2^-, \lambda_3^-, \lambda_1^+, \lambda_2^+, \lambda_3^+ = 0$

After first loop of step 2, some dual variables become active, i.e., their values are not equal to 0. In this case, $\lambda_1^- > 0, \lambda_2^- > 0, \lambda_3^- > 0$. Local optimal path flows $(f_1, f_2, f_3)$ and sub-optimal path flow $f_1$ and $f_3$ are found.

After Link Cost Update, the original algorithm starts with default dual value: $\lambda_1^-, \lambda_2^-, \lambda_3^-, \lambda_1^+, \lambda_2^+, \lambda_3^+ = 0$. Then, the original algorithm will have to repeat some computation on dual variables to re-obtain sub-optimal path flow $f_1$ and $f_3$.

With our primal-dual heuristics, the algorithm will approximately retain sub-optimal path flows after link cost update. In this case, the algorithm adjusts $\lambda_2^-$ and retain sub-optimal path flow $f_3$ for OD pair $O_2D_2$, while it also adjusts $\lambda_1^-$ and $\lambda_2^+$ and make path flows $f_1$ and $f_2$ as optimal as possible after link cost update.

After a new path ($f_4$ in this case) is generated, the original algorithm starts with default dual values: $\lambda_1^-, \lambda_2^-, \lambda_3^-, \lambda_1^+, \lambda_2^+, \lambda_3^+ = 0$. The original algorithm will repeat some computation to obtain sub-optimal path flows $f_1, f_3$.

With our primal-dual heuristics, the algorithm will adopt dual variable $(\lambda_1^-, \lambda_3^-)$ from previous computation and retains the sub-optimal path flows ($f_1, f_3$) according to equation (10) without further computation. The path flow optimization process will then focus only on the calculation of $f_2$ and $f_4$.

![Diagram](image)

Figure 2. Original Algorithm’s Dual Value Updating vs. The Primal-Dual Heuristic
As seen from Fig. 2, the LPFE solution algorithm starts with free flow traveling cost. After completing the first loop in Step 2, active dual variables $(\lambda_{a1}, \lambda_{a2}, \lambda_{a3})$ with local optimal path flows $(f_1, f_2, f_3)$ and sub-optimal path flow $f_1, f_3$ (See definition 1 & 2) are found. The original algorithm will update link costs with flow pattern $(f_1, f_2, f_3)$ and re-initialize all dual variables with default values $(0)$ to repeat the loop in Step 2. It needs to recalculate $(\lambda_{a1}, \lambda_{a3})$ to obtain sub-optimal path flows $f_1, f_3$ again. But by using the complementary relationship between dual variables $(\lambda_{a1}, \lambda_{a3})$ and their link cost $(C_{a1}, C_{a2})$, our primal-dual heuristic, to be presented in more detail later, will simply adjust active dual variable $\lambda_{a3}$ for the initialization process in Step 2 to retain sub-optimal path flow $f_3$ without further computation. The primal-dual heuristic also adjusts dual variables $(\lambda_{a1}, \lambda_{a2})$ to make path flows $(f_1, f_2)$’s initial value obtained by (10) as optimal as possible.

The Primal-Dual Heuristic

Based on earlier discussions, we propose the following primal-dual heuristic for updating the dual variables:

1) Initializing with default values 0 for all dual variables only for the first time Step 2 is implemented.
2) When link costs are fixed, find new path flows according to equation (10); If any constraint (2)-(6) is not satisfied, update dual variables according to the following heuristics:

Set maximum adjustment of all dual variables $\xi = 0$.

- For each link $a \in A_u$, compute $e = \ln x_a - \ln C_a$
  If $e \geq 0$, set $\mu_a = \mu_a + e$; otherwise, if $\mu_a > 0$, set $\mu_a = \max (\mu_a + e, 0)$.
  Update $\xi$ according to the value change on $\mu_a$.
- For each link $a \in A_o$, compute $e = \ln x_a - \ln v_a(1 + e_a)$
  If $e \geq 0$, set $\lambda_a = \lambda_a + e$; otherwise, if $\lambda_a > 0$, set $\lambda_a = \max (\lambda_a + e, 0)$.
  Update $\xi$ according to the value change on $\lambda_a^+$.
- For each link $a \in A_o$, compute $e = \ln v_a(1 - e_a) - \ln x_a$
  If $e \geq 0$, set $\lambda_a^+ = \lambda_a^+ + e$; otherwise, if $\lambda_a^+ > 0$, set $\lambda_a^+ = \max (\lambda_a^+ + e, 0)$.
  Update $\xi$ according to the value change on $\lambda_a^-$.
- For each O-D pair $rs$ for which the target upper bound information exists, compute
  
  $e = \ln \sum_{k \in K_{rs}} f_{rs}^{rs} - \ln D_{rs} \cdot (1 + e_{rs})$

  If $e \geq 0$, set $\varphi_{rs}^+ = \varphi_{rs}^+ + e$; otherwise, if $\varphi_{rs}^+ > 0$, set $\varphi_{rs}^+ = \max (\varphi_{rs}^+ + e, 0)$.
  Update $\xi$ according to the value change on $\varphi_{rs}^+$.
- For each O-D pair $rs$ for which the target lower bound information exists, compute
  
  $e = \ln D_{rs} \cdot (1 - e_{rs}) - \ln \sum_{k \in K_{rs}} f_{rs}^{rs}$

  If $e \geq 0$, set $\varphi_{rs}^- = \varphi_{rs}^- + e$; otherwise, if $\varphi_{rs}^- > 0$, set $\varphi_{rs}^- = \max (\varphi_{rs}^- + e, 0)$.
  Update $\xi$ according to the value change on $\varphi_{rs}^-$. 

TRB 2013 Annual Meeting  
Paper revised from original submittal.
Tang and Zhang

If $\xi < \eta_1$, update link costs, dual variables and objective function values according to procedure 3); otherwise, go back to the beginning of 2).

3) Apply the line search procedure to find path flow patterns that reduce the objective function value most, using the old path flow pattern in the previous loop and the new path flow pattern obtained in 2). Then, update link costs according to the line search path flow results. When link costs are updated link by link, use the following heuristics ($H_1$) to update dual variables:

- For each link $a \in A_u$, compute $e = \bar{t}_a - t_a$
  
  If $e < 0$, set $\mu_a = \max\left(\mu_a + e, \frac{1}{a} \ln x_a - e - \frac{1}{a} \ln C_a, 0\right)$; if $e = 0$, keep $\mu_a$ unchanged;
  
  If $e > 0$ and $\mu_a \geq 0$, set $\mu_a = \max(\mu_a - e, 0)$.

- For each link $a \in A_o$, compute $e = \bar{t}_a - t_a$
  
  If $e < 0$, set $\lambda_a^+ = \max\left(\lambda_a^+ + e, \frac{1}{a} \ln x_a - e - \frac{1}{a} \ln v_a(1 + \varepsilon_a), 0\right), \lambda_a^- = \max(\lambda_a^- + e, 0)$;
  
  Otherwise, if $e = 0$, keep $\lambda_a^+$ and $\lambda_a^-$ unchanged.

  If $e > 0$, set $\lambda_a^+ = \max(\lambda_a^+ - e, 0), \lambda_a^- = \max(\lambda_a^- - e, \frac{1}{a} \ln v_a(1 - \varepsilon_a) + e - \frac{1}{a} \ln x_a, 0)$;

  Otherwise, if $e = 0$, keep $\lambda_a^+$ and $\lambda_a^-$ unchanged.

- Keep $\phi_{r,s}^+, \phi_{r,s}^-$ unchanged during the link cost update process.

- Calculate the comparative gap between the new objective function value obtained from the line search procedure and the old objective function value obtained in the previous loop. If the comparative gap is lower than $\eta_2$, terminate; otherwise, go back to 2) with current link costs and dual variable values.

**Proposition 1.** The above primal-dual heuristic retains some sub-optimal path flows and their corresponding link flows once found during the updating process for dual variables.

**Proof.**

Without loss of generality, we assume the procedure finds the following sub-optimal path flow pattern: $f_{1}^{r,s}, f_{2}^{r,s}, \ldots, f_{k_r}^{r,s'}, K' \subseteq R_r^{r,s'}, R_r^{r,s'} \subseteq K_r^{r,s'}, r, s' \in D^*, D^* \subseteq D$ at certain iteration before termination with stable link cost pattern $C_a, a \in A_D^*$. (Here, stable link cost pattern means $C_a$ becomes constant or almost constant after a certain number of iterations).

According to the definition of sub-optimal path flows, there must be some dual variables ($\lambda_a^+, \lambda_a^-, \mu_a^+, \mu_a^-, \phi_{r,s}^+, \phi_{r,s}^-$, $a^* \in A_D^*, r, s' \in D^*$) satisfying the following equation:
Tang and Zhang

\[
 f^{RS}_{k'} = \exp \left( -\theta \sum_{a^* \in A_D^*} C_{a^*} \cdot \delta^{RS}_{k'a^*} - \theta \sum_{a^* \in M_D^*} \sum_{r \in D^{rs}} \sum_{k'} \delta^{RS}_{k'ra^*} \cdot \lambda^+_{a^*} 
 + \theta \sum_{a^* \in M_D^*} \sum_{r \in D^{rs}} \sum_{k'} \delta^{RS}_{k'ra^*} \cdot \lambda^-_{a^*} - \theta \sum_{a^* \in M_D^*} \sum_{r \in D^{rs}} \sum_{k'} \delta^{RS}_{k'ra^*} \cdot \mu_{a^*} 
 - \theta \phi^{RS*}_{k'} + \theta \phi^{RS}_{k'} \right) \tag{16}
\]

Follow the above primal-dual heuristic, we would know that \( \lambda^+_{a^*}, \lambda^-_{a^*}, \mu_{a^*}, \phi^{RS*}_{a}, \phi^{RS}_{a} \), \( a^* \in A_D^* \), \( rs \in D^* \) will be updated to compensate the link cost change as much as possible to maintain the equality between both sides of equation (16). If sub-optimal path flows with stable link costs were found, the above primal-dual heuristic will keep the equality exactly between both sides of equation (16) after the link cost updating procedure \( (H_1) \). With any sub-optimal path flow pattern \( \{ f^{RS}_{k'}, f^{RS}_{k''}, ..., f^{RS}_{k'''} \}, k' \in K^{RS}, k'' \subseteq K^{RS}, r'' \in D^*, D^* \subseteq D \) found, so is their corresponding link flow pattern \( (x_a = \sum_{rs \in D^*} \sum_{k \in K^{RS}} f^{RS}_{k}, a \in A_D^* \)\). The line search procedure for global convergence won’t change those links’ flow pattern, which then brings no change to those links’ costs and further no effect on their corresponding dual variables according to the primal-dual heuristic. So, the equality between both sides of Equation (16) is retained with no additional computation cost after performing the line search procedure. Also, the new path generation procedure doesn’t change link costs. If any sub-optimal path flow pattern was found, those sub-optimal path flows are intact after performing the new path generation procedure according to Equation (16) and the primal-dual heuristic. \( Q.E.D. \)

Remark 1:

The above primal-dual heuristic is like a warm-start technique in regular optimization. It aims at providing a good starting point for updating dual variables after link costs change.

Without loss of generality, we assume the algorithm finds a set of paths whose estimated volumes are very close to their corresponding sub-optimal values at a certain iteration before termination. We define link set \( A_D^{**} \) as those links that are crossed by these paths. A subset of them \( A_D^{**} \subset A_D^{**} \) have minor link cost changes in the following link cost updating, while others \( a^* \notin A_D^{**} \) have no link cost change in the following link cost updating. Among the links belonging to \( A_D^{**} \), \( A_M^{**} \) denotes the measured link set, \( A_U^{**} \) denotes the unmeasured link set.

After the link cost updating procedure \( (H_1) \), the new flow volumes for those paths are obtained by the following equation according to (10):
Tang and Zhang

\[ f_{rs}^{rs} = \exp \left( -\theta \sum_{a^* \in A_D^{**}, \text{but } a^* \in A_D^{**}} C_a \cdot \delta_{ka}^{rs} - \theta \sum_{a^* \in A_M^{**}, \text{but } a^* \in A_M^{**}} \sum_{rs, k} \delta_{ka}^{rs} \cdot \lambda_a^+ + \theta \sum_{a^* \in A_M^{**}, \text{but } a^* \in A_M^{**}} \sum_{rs, k} \delta_{ka}^{rs} \cdot \lambda_a^- \right) \]

\[ \times \exp \left( -\theta \sum_{a^* \in A_D^{**}, \text{but } a^* \in A_D^{**}} C_a^{new} \cdot \delta_{ka}^{rs} - \theta \sum_{a^* \in A_M^{**}, \text{but } a^* \in A_M^{**}} \sum_{rs, k} \delta_{ka}^{rs} \cdot new\lambda_a^+ \right) \]

\[ \times \exp \left( +\theta \sum_{a^* \in A_M^{**}, \text{but } a^* \in A_M^{**}} \sum_{rs, k} \delta_{ka}^{rs} \cdot new\lambda_a^- \right) \]

(17)

In the first half of Equation (17), the link cost and dual variables don’t change at all after performing link cost updating procedure (H₁), which brings no change to the new path flow \( f_{rs}^{rs} \) while, in the second half of Equation (17), the changes on dual variables will compensate the minor link cost change as much as possible, which finally makes the initial path flow \( f_{rs}^{rs} \) almost unchanged. This new initial path flow \( f_{rs}^{rs} \) is usually much closer to its sub-optimal value compared with the ones which are obtained by default dual variable values (0). The reason is that fewer link and historical OD demand constraints (2)-(6) will be active in the final solution compared with the huge set of available paths, most of whom only carries negligible flows. Once the algorithm finds those limited number of active constraints with their corresponding dual value estimates to represent some major path flow patterns (each path bears almost 30%-90% of target OD pair demand) close to their sub-optimal value, those major paths’ flow patterns would not be affected much by link cost updating or new path generation. The rest of algorithm actually works on finding new marginal paths, which bear 10% or lower of target OD pair demand, and adjusts their volumes to achieve global optimal.

The situation is more obvious in large networks, under which the ratio of active constrains to the set of available paths between OD pairs is much lower. Hundreds of paths will carry negligible amount of flow in the final solution. These path flows are not the key elements to affect the objective function value. The original algorithm spends quite an amount of computational time to balance those negligible path flows with their corresponding link volume measurements and OD demand observation. The primal dual heuristic helps the algorithm memorize and duplicate major path flow patterns after changes in link costs or the generation a new path set.
Remark 2:

In practice, it’s impossible to obtain link flow counts and historical OD demand with a hundred percent accuracy. Many researchers (such as Lam and Lo [29], Chootinan et al. [24]) suggest using user-specific measurement error bounds to handle the data inconsistency caused by inaccurate data. But there is no uniform way to set up user-specific error bounds. It’s common to follow the so called “try and go” procedure to set up those error bounds. But the original algorithm, which could only re-estimate new path flows from default dual variable values (0) after any link flow count or historical OD demand error bound change, causes tremendous computation burden for this “try and go” procedure. Now through the primal-dual heuristic, the algorithm could establish some connections between new solutions that comply with new error bounds with old solutions that are optimized with old error bounds. The algorithm could treat the old path flow pattern as some intermediate result, which holds some sub-optimal path flow patterns. The reason is that changes in some link flow counts’ error bounds would not affect sub-optimal paths that do not go through these links. So the algorithm could start with the old path flow pattern and its corresponding dual variable values to find the new path flow patterns, within which some of original sub-optimal path flows will be kept intact through the primal-dual heuristic. The new algorithm only makes necessary flow adjustments on paths that hold different flow patterns to represent the new observation information.

Numerical Examples

Two example networks (Figure 2) are used to evaluate the primal-dual heuristic. In the first example, the highway network of St. Helena, a small town located in the center of Napa Valley, California, is used to represent small-size networks. It consists of 226 one-way links, 88 of which have link traffic counts (Details shown in Table 1). In the second example, the highway network of Sacramento, the capital city of California, is used to represent large-size networks. The Sacramento network consists of 8510 one-way links, 70 of which have link traffic counts (Details shown in Table 1). The link traffic counts in cities of St. Helena and Sacramento are derived from tube and loop counts in the morning peak hour. We collected average morning peak hour tube and loop counts during different weekdays of a week. Then, we made some random adjustment on several observed link counts to generate the link count observation data.
Figure 2. St. Helena (Left) and Sacramento (Right) Transportation Network

<table>
<thead>
<tr>
<th>Network Name</th>
<th>Number of Nodes</th>
<th>Number of Links (One-way Per Link)</th>
<th>Number of Measured Links</th>
<th>Number of Target OD Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>St. Helena</td>
<td>86</td>
<td>226</td>
<td>88</td>
<td>756</td>
</tr>
<tr>
<td>Sacramento</td>
<td>2554</td>
<td>8510</td>
<td>70</td>
<td>14042</td>
</tr>
</tbody>
</table>
We applied our primal-dual heuristic LPFE solution algorithm to the St. Helena network with the given traffic counts and some arbitrarily added error bounds. The computational results are summarized in Figure 3, which depicts the decrease of comparative gap of the objective function value in each major and sub-iteration (indicated by a dot or square) and the amount of time taken in each iteration (which is measured by the horizontal spacing between each pair of dots/squares). Here the major iterations are those concerning path generation and line search, and sub-iterations are those concerning the updating of the dual variables. The reason we use the comparative gap (See Appendix for the definition) rather than the actual objective function values is that for different sizes of networks their objective function values range widely and can be hard to compare. The comparative gap normalizes their values and makes it easy to compare algorithm performance for different sizes of networks.

As one can see from Fig. 3, during the first several iterations, the primal-dual heuristic algorithm performs similarly as the original algorithm in terms of reducing the comparative gap. But each of the primal-dual heuristic algorithm’s major iteration takes less computational time. The reason for this is obvious. At first several major iterations, the number of sub-optimal paths are fewer. Due to the lack of necessary paths, the heuristic, which makes use of previous dual variables, does not have a clear advantage in reducing objective function values, but it does speed up the path generation process. Once enough paths are generated, the algorithm converges much more quickly than the original algorithm, due to its reuse of sub-optimal paths and dual values from previous iterations. In contrast, the original algorithm stalls after more paths are generated, because without using the values of dual variables from previous iterations it does many redundant computations.
The convergence patterns (Fig. 4) from the Sacramento network is similar with that from the St. Helena network (Fig. 3) but showing greater computational advantage of the primal-dual heuristic. The new algorithm enables quicker drops in the objective function value even at the first several major iterations than the original algorithm. The reason is that when the network size increases, the number of potential paths increases dramatically, but the number of active constraints are not expanding as quickly; the primal-dual heuristic helps the algorithm retain those dual variables associated with the few active constraints and find their corresponding sub optimal path flows more quickly; Thus for each new path generated, its contribution to the reduction of the objective function value is substantial before reaching a small neighborhood of the optimal solution. In contrast, the original algorithm searches for a larger set of paths, some of which contain negligible flow that contribute little to the reduction of the objective function value. It repeatedly adds and drops such paths, which consumes a large share of computational time in the algorithm hence considerably slows the convergence of the algorithm.
Conclusions

In this paper, we introduce a primal-dual heuristic for path flow estimation using traffic counts and historical OD demands. The key element of this heuristic is to use the complementary relation between link flow and dual variables to “warm” start the updating of the dual variables. Two examples, one small network and one large network, are used to demonstrate the effectiveness of the proposed heuristic and it was found that the proposed algorithm converges much more quickly than the original solution algorithm, particularly in the case of the large network.

Reference

Tang and Zhang

Appendix, Notations:

$f_k^{rs}$: Estimated path $k$'s flow between origin $r$ and destination $s$

$\theta$: Dispersion parameter

$A$: Set of all links

$D, D_{rs}$: Target OD pairs and historical demand of OD pair $rs$

$D^*$: Any subset of Target OD pairs

$M, U$: Set of measured links (with traffic counts) and unmeasured links (without traffic counts)

$K_{rs}$: Set of paths between origin $r$ and destination $s$

$\varepsilon_a, \varepsilon_{rs}$: Relative measurement errors $[0,1]$ for traffic count on link $a \in M$ and historical travel demand of OD pair $rs \in D$

$v_a, x_a$: Observed flow and estimated flow on link $a$

$C_a, t_a(\cdot)$: Capacity and cost function of link $a$

$\delta^{rs}_{ka}$: Path link incidence indicator: 1 if link $a$ is on path $k$ between origin $r$ and destination $s$, and 0 otherwise

$\lambda^+_a, \lambda^-_a$: Dual variables associated with upper and lower bounds of the link flow observation constraint on link $a \in M$;

$\mu_a$: Dual variables associated with link capacity constraint on link $a \in U$

$\varphi^+_{rs}, \varphi^-_{rs}$: Dual variables associated with upper and lower bounds of the historical travel demand constraint on OD pair $rs \in D$;

$A_D$: Set of links consisted by paths $k \in K^{rs}, rs \in D^*$

$M_D, U_D$: Set of measured and unmeasured links by paths $k \in K^{rs}, rs \in D^*$

$H_1$: Primal-Dual Heuristic during link cost updating procedure

$Z(f)$: Objective function value

$\eta_1$: Maximum allowable dual value change

$\eta_2$: Minimum comparative objective function value change

$z_t$: Objective function value obtained at the end of iteration $t$. The algorithm updates objective function value at the end of Step 2 (minor iteration) or Step 3 (major iteration).
z_{\text{min}}: \text{Minimum objective function value obtained from the algorithm}

\text{Gap}(z_t): \text{Comparative gap of objective function value}

\text{Gap}(z_t) \triangleq \frac{z_t - z_{\text{min}}}{z_{\text{min}}} \times 100\%