Asia-Pacific Journal of Operational Research
Vol. 25, No. 2 (2008) 135–150

ON SOLVING SHORTEST PATHS WITH A LEAST-SQUARES PRIMAL-DUAL ALGORITHM

I.-LIN WANG

Department of Industrial and Information Management
National Cheng Kung University, Tainan, Taiwan
ilinwang@mail.ncku.edu.tw

Received 15 August 2005
Accepted 17 July 2007

Recently a new least-squares primal-dual (LSPD) algorithm, that is impervious to degeneracy, has effectively been applied to solving linear programming problems by Barnes *et al.*, 2002. In this paper, we show an application of LSPD to shortest path problems with nonnegative arc length is equivalent to the Dijkstra's algorithm. We also compare the LSPD algorithm with the conventional primal-dual algorithm in solving shortest path problems and show their difference due to degeneracy in solving the 1-1 shortest path problems.

Keywords: Least-squares; primal-dual algorithm; shortest path; Dijkstra's algorithm.

1. Introduction

The *least-squares primal-dual algorithm* (LSPD) (Barnes *et al.*, 2002) is a primal-dual algorithm for solving LPs. Instead of minimizing the sum of the absolute infeasibility in the constraints when solving the restricted primal problem (RPP), as does the original primal-dual algorithm (PD), LSPD tries to minimize the sum of the squares of the infeasibility.

In particular, to solve an LP:

$$\min cx$$

$$\text{s.t.} \quad Hx = b,\ x \geq 0$$

with an initial feasible dual solution π, LSPD maintains complementary slackness conditions by seeking solutions to a quadratic RPP which is a *non-negative least-squares problem* (NNLS):

$$\min \|b - Ex\|^2 \tag{1.1}$$

$$\text{s.t.} \quad x \geq 0$$

where $E = \{H_{.j} : \pi H_{.j} = c_j\}$. Using the solution x^* to NNLS, LSPD identifies a dual improving direction $s^* = b - Ex^*$ and calculates the step size θ to obtain

an improved dual solution $\pi + \theta s^*$. LSPD then identifies new column set E using the updated dual solutions. These procedures are repeated until $s^* = 0$, which means primal feasibility (and thus optimality) has been attained. Solving NNLS usually requires the solving of normal equations $E^T Ex = E^T b$ by Cholesky or QR factorizations. When solving min-cost network flow problems, the normal equations can be solved through a specialized combinatorial implementation of LSPD (see Gopalakrishnan, 2002; Barnes *et al.*, 2005) without any matrix inversion procedures.

LSPD improves the dual solution in a nondegenerate way shown to be more efficient than the Hungarian method (a PD algorithm) in solving assignment problems (Barnes *et al.*, 2005). Thus LSPD is more efficient than PD for this special class of network flow problems. In this paper, we consider another class of network flow problems, the shortest path problems with nonnegative arc lengths. We propose a simplified LSPD implementation to solve ALL-1 shortest path problems (ALL-1-SP) and 1-1 shortest path problems (1-1-SP) on graphs with nonnegative arc lengths, where ALL-1-SP computes a shortest path tree from each node to a specific sink node, and 1-1-SP computes a shortest path for an origin-destination pair. Without loss of generality, in this paper, we assume the ALL-1-SP and 1-1-SP are always primal feasible. These shortest path problems are usually solved by Dijkstra's algorithm (Dijkstra, 1959), which can also be viewed as an application of PD (Padadimitriou and Steiglitz, 1982). We show our LSPD implementation is identical to Dijkstra's algorithm in the sense that both algorithms grow the same shortest path, and give more insights into LSPD and PD for solving these shortest path problems. In solving 1-1-SP, we show that PD may give degenerate pivots that slows down its performance, while LSPD will always give the same nondegenerate pivots as does by Dijkstra's algorithm. Such a degeneracy effect of PD can also be verified from our preliminary computational experiments.

The paper is organized as follows. In Sec. 2, we discuss how LSPD, PD and Dijkstra's algorithm solve ALL-1-SP and show their equivalence in solving ALL-1-SP. Section 3 illustrates steps of LSPD in solving 1-1-SP and then compares it with PD and Dijkstra's algorithm. Section 4 provides computational evidence on the degeneracy effect of PD in solving 1-1-SP. We give conclusions in Sec. 5.

2. Solving the ALL-1 Shortest Path Problem

For a digraph $G = (N, A)$ with n nodes and m arcs where N and A denote the set of nodes and arcs respectively, if we let c_{ij} be the length of arc (i, j), then we can formulate an ALL-1-SP as an LP:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} = Z^{P*}_{ALL-1}(x) \tag{2.1}$$

$$\text{s.t.} \quad \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = b_i = \begin{cases} 1, & \text{if } i \neq t \\ -(n-1), & \text{if } i = t \end{cases} \quad \forall\, i \in N \tag{2.2}$$

$$x_{ij} \geq 0 \quad \forall\, (i,j) \in A.$$

Specifically, the problem can be viewed as if every node other than t sends one unit of flow to satisfy the demand $(n-1)$ of node t in a way that minimizes the total transportation cost. The constraint coefficient matrix of (2.2) is the node-arc incidence matrix of G, and contains a redundant constraint. Without loss of generality, we can remove the last row of (2.2) to get a new coefficient matrix $\widehat{H}$ and obtain the following LP:

$$\min cx = Z^{P*}_{ALL-1}(x) \qquad \text{(ALL-1-Primal)}$$

$$\text{s.t.} \quad \widehat{H}x = 1 \quad \forall\, i \in N \backslash t \qquad (2.3)$$

$$x \geq 0,$$

whose dual is

$$\max \sum_{i \in N \backslash t} \pi_i = Z^{D*}_{ALL-1}(\pi) \qquad \text{(ALL-1-Dual)}$$

$$\text{s.t.} \quad \pi_i - \pi_j \leq c_{ij} \quad \forall\, (i,j) \in A, i, j \neq t \qquad (2.4)$$

$$\pi_i \leq c_{it} \quad \forall\, (i,t) \in A \qquad (2.5)$$

$$-\pi_j \leq c_{tj} \quad \forall\, (t,j) \in A. \qquad (2.6)$$

These LPs can be solved by the network simplex algorithm (Ahuja *et al.*, 1993), where a basis corresponds to a spanning tree, the dual variable π_i for each node i corresponds to a distance label from node i to node t (thus $\pi_t = 0$), and the reduced cost c^{π}_{ij} for each arc (i,j) is $c_{ij} - \pi_i + \pi_j$.

Given an initial feasible dual solution π (we can use $\pi = 0$ because $c_{ij} \geq 0$ for each arc (i,j)) for ALL-1-Dual, we first identify a set of *admissible arcs*, denoted by $\widehat{A} = \{(i,j) \in A : c^{\pi}_{ij} = 0\}$. Let $\widehat{G} = (N, \widehat{A})$ denote the *admissible graph*, which contains all the nodes in N but only arcs in $\widehat{A}$. We call a node i an *admissible node* if there exists a path from i to t in $\widehat{G}$, or $i = t$. Thus the admissible node set $\widehat{N}$ is the connected component of $\widehat{G}$ that contains t.

LSPD solves the restricted primal problem, which seeks the flow assignment x^* on admissible graph $\widehat{G}$ that minimizes the sum of squares of the node imbalance (or slackness vector) $\delta = b - \widehat{E}x_{\widehat{A}}$:

$$\min \sum_{i \in N \backslash t} \delta_i^2 = \sum_{i \in N \backslash t} \left(b_i - \sum_{a \in \widehat{A}} \widehat{E}_{ia} x_a \right)^2 \qquad \text{(NNLS-RPP)}$$

$$\text{s.t.} \quad x_a \geq 0 \quad \forall\, a \in \widehat{A},$$

where $\widehat{E} = \{\widehat{H}_{\cdot a} : \pi \widehat{H}_{\cdot a} = c_a,\ \forall\, a \in A\}$ corresponds to the arcs in $\widehat{A}$. Note that $x_a = 0$ for each nonadmissible arc $a \in A \diagdown \widehat{A}$.

Problem NNLS-RPP is an instance of NNLS and can be solved by the algorithm of Leichner *et al.* (1993). LSPD uses the optimal imbalance δ^* to NNLS-RPP to improve π (see Gopalakrishnan, 2002; Barnes *et al.*, 2005, for the proof). Here we

Algorithm 2.1. LSPD-ALL-1

begin
Initialize: $\forall$ node i, $\pi_i := 0$; add node t to $\widehat{N}$;
Identify admissible arc set $\widehat{A}$ and admissible node set $\widehat{N}$;
while $\left|\widehat{N}\right| < n$ ***do***
$\delta^* = NNLS\text{-}ALL\text{-}1(\widehat{G}, \widehat{N})$;
Let $\widetilde{A} = \{(i,j) \in A : \delta_i^* > \delta_j^*\}$, $\theta = \min_{(i,j)\in\widetilde{A}} \{\frac{c_{ij}^{\pi}}{\delta_i^* - \delta_j^*}\}$;
$\pi = \pi + \theta\delta^*$;
Identify admissible arc set $\widehat{A}$ and admissible node set $\widehat{N}$;
end

Procedure NNLS-ALL-1$(\widehat{\mathbf{G}}, \widehat{\mathbf{N}})$
begin
for $i = 1$ to n ***do***
if node $i \in \widehat{N}$ ***then***
$\delta_i^* = 0$;
else
$\delta_i^* = 1$;
return δ^*;
end

give a specialized implementation (Algorithm 2.1 *LSPD-ALL-1*) to solve ALL-1-SP. It contains procedure *NNLS-ALL-1* for solving NNLS-RPP.

Figure 1 illustrates how algorithm *LSPD-ALL-1* solves an ALL-1 shortest path problem.

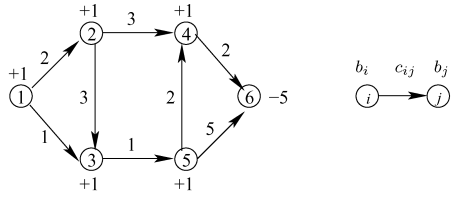
Now we show that *LSPD-ALL-1* correctly computes an ALL-1 shortest tree.

Theorem 2.1. *The δ^* computed by the procedure NNLS-ALL-1 solves problem NNLS-RPP.*

Proof. For each nonadmissible node, since it has no path of admissible arcs to t, its optimal imbalance remains 1. On the other hand, each admissible node can always ship its imbalance to t via uncapacitated admissible arcs so that its optimal imbalance becomes zero. Therefore the δ^* computed by procedure *NNLS-ALL-1* corresponds to the optimal imbalance δ_i^* for NNLS-RPP. □

Lemma 2.1. *Algorithm LSPD-ALL-1 solves the ALL-1 shortest path problem.*

Proof. By Theorem 2.1, δ^* solves quadratic RPP (i.e. NNLS-RPP), so δ^* is a dual ascent direction (Gopalakrishnan 2002; Barnes *et al.*, 2002). *LSPD-ALL-1* iteratively computes the step length θ to update dual variables π, identifies admissible



Iteration 1: Dual: $\pi = [0,0,0,0,0]$ $\widehat{A} = \emptyset$, $\widehat{N} = \{6\}$
RPP: $min\ 1^2 + 1^2 + 1^2 + 1^2 + 1^2$
$\delta^* = [1,1,1,1,1]$
$\widetilde{A} = \{(4,6),(5,6)\}$ $\theta = min\{2,5\} = 2$

Iteration 2: Dual: $\pi = [2,2,2,2,2]$ $\widehat{A} = \{((4,6)\}$, $\widehat{N} = \{6,4\}$
RPP: $min\ 1^2 + 1^2 + 1^2 + (1-x_{46})^2 + 1^2$
$s.t.$ $x_{46} \geq 0$
$\delta^* = [1,1,1,0,1]$, $x^*_{46} = 1$
$\widetilde{A} = \{(2,4),(5,4),(5,6)\}$ $\theta = min\{3,2,3\} = 2$

Iteration 3: Dual: $\pi = [4,4,4,2,4]$ $\widehat{A} = \{((4,6),(5,4)\}$, $\widehat{N} = \{6,4,5\}$
RPP: $min\ 1^2 + 1^2 + 1^2 + (1-x_{46}+x_{54})^2 + (1-x_{54})^2$
$s.t.$ $x_{46}, x_{54} \geq 0$
$\delta^* = [1,1,1,0,0]$, $x^*_{46} = 2, x^*_{54} = 1$
$\widetilde{A} = \{(2,4),(3,5)\}$ $\theta = min\{1,1\} = 1$

Iteration 4: Dual: $\pi = [5,5,5,2,4]$ $\widehat{A} = \{((4,6),(5,4),(2,4),(3,5)\}$, $\widehat{N} = \{6,4,5,2,3\}$
RPP: $min\ 1^2 + (1-x_{24})^2 + (1-x_{35})^2 + (1-x_{46}+x_{54})^2 + (1-x_{54}+x_{35})^2$
$s.t.$ $x_{24}, x_{35}, x_{46}, x_{54} \geq 0$
$\delta^* = [1,0,0,0,0]$, $x^*_{24} = x^*_{35} = 1, x^*_{46} = 4, x^*_{54} = 2$
$\widetilde{A} = \{(1,2),(1,3)\}$ $\theta = min\{2,1\} = 1$

Iteration 5: Dual: $\pi = [6,5,5,2,4]$ $\widehat{A} = \{((4,6),(5,4),(2,4),(3,5),(1,3)\}$, $\widehat{N} = \{6,4,5,2,3,1\}$
$|\widehat{N}| = |N|$, Algorithm *LSPD-ALL-1* terminated.

Fig. 1. A small ALL-1 shortest path example solved by algorithm *LSPD-ALL-1*.

arcs (i.e. columns), and solves NNLS-RPP. Assuming each node $i \in N \backslash t$ reaches t, *LSPD-ALL-1* gives the following remarks:

(a) $\delta^*_i = 0$, $\forall i \in \widehat{N}$ and $\delta^*_i = 1$, $\forall i \in N \backslash \widehat{N}$.
(b) Let $\widehat{N}^k$ denote the set of $\widehat{N}$ in the beginning of iteration k, then $\widehat{N}^k \subseteq \widehat{N}^{k+1}$ and $|\widehat{N}^{k+1}| \geq |\widehat{N}^k| + 1$
(c) In at most $n-1$ major iterations, the algorithm *LSPD-ALL-1* terminates with $\widehat{N} = N$. Thus finally when $\sum_{i \in N \backslash t} \delta^{*2}_i$ vanishes, which means the primal feasibility has been attained, and since the dual feasibility and complementary slackness conditions are maintained during the whole procedure, *LSPD-ALL-1* solves ALL-1-SP. □

Now we compare algorithm *LSPD-ALL-1* with the original primal-dual algorithm to solve the ALL-1 shortest path problem. The only difference between algorithm LSPD and PD is that they solve different RPPs. PD solves the

following RPP:

$$\min \sum_{i\in N\setminus t} \delta_i = \sum_{i\in N\setminus t} \left(1 - \sum_{a\in \widehat{A}} \widehat{E}_{ia} x_a\right)$$
$$\text{s.t.} \quad x_a \geq 0 \quad \forall\, a \in \widehat{A},\ \delta_i \geq 0 \quad \forall\, i \in N\setminus t,$$

whose dual is

$$\begin{aligned} \max \sum_{i\in N\setminus t} \rho_i & & \text{(DRPP-ALL-1)} \\ \text{s.t.} \quad \rho_i \leq \rho_j, & \quad \forall\,(i,j) \in \widehat{A},\ i,j \neq t \\ \rho_i \leq 0, & \quad \forall\,(i,t) \in \widehat{A} \\ \rho_j \geq 0, & \quad \forall\,(t,j) \in \widehat{A} \\ \rho_i \leq 1, & \quad \forall\, i \in N\setminus t. \end{aligned}$$

The optimal dual solution ρ^* of DRPP-ALL-1 is used as a dual-ascent direction. For each node i that cannot reach t along admissible arcs in $\widehat{A}$ (i.e. i is nonadmissible), it is easy to observe that $\rho_i^* = 1$. Also, if node i is admissible, that is, $i = t$ or there exists a path from i to t with intermediate nodes $\{i_1, i_2, i_3, \ldots, i_k\}$, then $\rho_{i_1}^* = \rho_{i_2}^* = \rho_{i_3}^* = \cdots = \rho_{i_k}^* = 0$. In other words, PD will have $\rho^* = 0$ for all the admissible nodes, and $\rho^* = 1$ for all the nonadmissible nodes. Thus the improving direction ρ^* obtained by the PD is identical to the one obtained by LSPD.

Therefore we can say that LSPD and PD are identical to each other in solving ALL-1-SP since they produce the same improving direction and step length and also construct the same restricted network $\widehat{G}$ at each iteration.

Now let us compare LSPD and PD with Dijkstra's algorithm. Since Dijkstra's algorithm is usually stated as a 1-ALL shortest path algorithm, for our convenience, we construct a new graph $G'' = (N, A'')$ by reversing all the arc directions of A so that an ALL-1 shortest path problem on G to sink t becomes a 1-ALL shortest problem from source t on G''. Initialize a node set V as empty and its complement $\overline{V}$ as the whole node set N. The distance label for each node i, denoted as $d(i)$, represents the distance from t to i in G''. Define $pred(j) = i$ if node i is the predecessor of node j.

We say a node is *permanently labeled* if it is put into V. A node is *labeled* if its distance label is finite. A node is *temporarily labeled* if it is labeled but not permanently labeled.

Dijkstra's algorithm starts by permanently labeling t, and then iteratively labels temporary nodes with arcs from permanently labeled nodes. This is identical to *LSPD-ALL-1*, which grows admissible nodes only from admissible nodes. In fact, in every major iteration, the set of admissible nodes in *LSPD-ALL-1* is the same as the set of permanently labeled nodes in Dijkstra.

Algorithm 2.2. Dijkstra(G'')

begin
Initialize: $\forall$ node $i \in N\backslash t$, $d(i) := \infty$, $pred(i) = -1$;
$d(t) := 0$, $pred(t) := 0$; $V := \emptyset, \overline{V} := N$;
while $|V| < n$ ***do***
let $i \in \overline{V}$ be a node such that $d(i) = \min\{d(j) : j \in \overline{V}\}$
$V := V \cup \{i\}; \overline{V} := \overline{V}\backslash\{i\}$
for each $(i, j) \in A''$ ***do***
if $d(j) > d(i) + c_{ij}$ ***then***
$d(j) := d(i) + c_{ij}$; $pred(j) := i$;
end

Theorem 2.2. *Both algorithm Dijkstra and LSPD-ALL-1 choose the same node to become permanently labeled (in Dijkstra) or admissible (in LSPD-ALL-1) in each major iteration.*

Proof. See Appendix. □

From this discussion, we conclude that when solving the ALL-1 shortest path problem with nonnegative arc lengths, all three algorithms, Dijkstra, *LSPD-ALL-1*, and PD, will perform the same operations in each iteration. In fact, this result is not surprising due to the nondegeneracy of the ALL-1-SP problem structure. In this ALL-1 shortest path problem, each node other than t has supply 1 to send to t. Thus in each iteration of LSPD and PD, the primal infeasibility will strictly decrease since a new admissible node can always be discovered; hence each pivot is always nondegenerate.

LSPD is an algorithm designed to take advantage of doing nondegenerate pivots in each iteration. Therefore, in this special case it performs just as efficiently as the other two algorithms. Next we will see that because the 1-1-SP does not have the nondegenerate property, in general *LSPD-1-1* does a better job than the original PD algorithm.

3. Solving the 1-1 Shortest Path Problem

Unlike the ALL-1 shortest path problem which searches for a shortest path tree rooted at node t, the 1-1 shortest path problem only asks for a shortest path from node s to node t. It can be viewed as finding the minimum cost way of sending a unit flow from s to t with the minimum cost via uncapacitated arcs. Its LP formulation is similar to the ALL-1 formulation except now node imbalance vector b only has two nonzeros: $+1$ for s, and -1 for t. Since an ALL-1-SP algorithm could be overkill when solving a 1-1-SP, here we develop specialized LSPD and PD implementations for solving 1-1-SP and then develop more insights into the behavior of LSPD, PD, and Dijkstra's algorithm.

Again, we remove the redundant row corresponding to t in the LP formulation, which gives us the following primal and dual formulations:

$$\min cx = Z_{1-1}^{P*}(x) \tag{1-1-Primal}$$

$$\text{s.t.} \quad \widehat{H}x = \begin{cases} 1, & \text{if } i = s \\ 0, & \text{if } i \in N \backslash \{s, t\} \end{cases} \quad \forall\, i \in N \backslash t \tag{3.1}$$

$$x \geq 0,$$

whose dual is

$$\max\ \pi_s = Z_{1-1}^{D*}(\pi) \tag{1-1-Dual}$$

$$\text{s.t.} \quad \pi_i - \pi_j \leq c_{ij}, \quad \forall\, (i,j) \in A,\ i,\ j \neq t \tag{3.2}$$

$$\pi_i \leq c_{it}, \quad \forall\, (i,t) \in A \tag{3.3}$$

$$-\pi_j \leq c_{tj}, \quad \forall\, (t,j) \in A \tag{3.4}$$

Here the right-hand-side of (3.1) only contains one nonzero (+1 for node s). This makes the dual objective $Z_{1-1}^{D*}(\pi)$ differ from that of ALL-1-SP, $Z_{ALL-1}^{D*}(\pi)$, in which $Z_{1-1}^{D*}(\pi)$ maximizes only π_s while $Z_{ALL-1}^{D*}(\pi)$ maximizes the sum $\sum_{i \in N \backslash t} \pi_i$.

First, we redefine an *admissible node* as a node that is reachable from s only via admissible arcs. We give a new procedure *NNLS-1-1* to solve NNLS-RPP in our 1-1 shortest path algorithm, *LSPD-1-1*, as shown in Algorithm 3.1. Algorithm *LSPD-1-1* can be remarked as shown in Algorithm 3.1.

Now let us show that *LSPD-1-1* correctly computes the shortest path from s to t.

Theorem 3.1. *The δ^* computed by the procedure NNLS-1-1 solves problem NNLS-RPP.*

Proof. NNLS-RPP is a quadratic programming problem. If we relax the nonnegativity constraints, it is a least-squares problem which can be solved by solving the normal equation $\widehat{E}^T \widehat{E} x^* = \widehat{E}^T b$. In other words, $\widehat{E}^T \delta^* = \widehat{E}^T (b - \widehat{E} x^*) = 0$. Note that each row of $\widehat{E}^T$ contains only two nonzero entries (i.e. +1 and −1) which represent an admissible arc. In other words, $\widehat{E}^T \delta^* = 0$ implies $\delta_i^* = \delta_j^*$ for each admissible arc (i, j) which implies all admissible nodes have the same optimal imbalance δ^* since $(\widehat{N}, \widehat{A})$ is connected. Since the total system imbalance is 1 (from the source s), the optimal least-squares solution δ_i^* for NNLS-RPP will be $\frac{1}{|\widehat{N}|}$ for each admissible node i. Using the optimal imbalance δ^*, it is easy to compute the unique optimal arc flow x^* and verify that $x^* \geq 0$ by traversing nodes on the component that contains the source node s (For more details in the application of LSPD on network problems, see Gopalakrishnan (2002); Barnes *et al.* (2005)). Thus the optimal imbalance δ^*, using the procedure *NNLS-1-1*, solves NNLS-RPP. □

Algorithm 3.1. LSPD-1-1

begin

Initialize: $\forall$ node i, $\pi_i := 0$; add node s to $\widehat{N}$;

Identify admissible arc set $\widehat{A}$ and admissible node set $\widehat{N}$;

while node $t \notin \widehat{N}$ ***do***

$\delta^* = NNLS\text{-}1\text{-}1(\widehat{G}, \widehat{N})$;

Let $\widetilde{A} = \{(i,j) \in A : \delta_i^* > \delta_j^*\}$, $\theta = \min_{(i,j)\in\widetilde{A}} \{\frac{c_{ij}^{\pi}}{\delta_i^* - \delta_j^*}\}$;

$\pi = \pi + \theta\delta^*$;

Identify admissible arc set $\widehat{A}$ and admissible node set $\widehat{N}$;

end

Procedure NNLS-1-1$(\widehat{\mathbf{G}}, \widehat{\mathbf{N}})$

begin

for $i = 1$ to n ***do***

if node $i \in \widehat{N}$ ***then***

$\delta_i^* = \frac{1}{|\widehat{N}|}$;

else

$\delta_i^* = 0$;

return δ^*;

end

Lemma 3.1. *Algorithm LSPD-1-1 solves the 1-1 shortest path problem from s to t.*

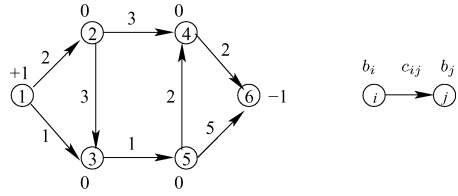
Proof. Using Theorem 3.1, δ^* solves NNLS-RPP. δ^* is a dual ascent direction (Gopalakrishnan, 2002; Barnes *et al.*, 2002). *LSPD-1-1* iteratively computes the step length θ to update dual variables π, identifies admissible arcs (i.e. columns), and solves NNLS-RPP. Assuming node s reaches t, *LSPD-1-1* gives the following remarks:

(a) $\delta_i^* = \frac{1}{|\widehat{N}|}$, $\forall\, i \in \widehat{N}$ and $\delta_i^* = 0$, $\forall\, i \in N \backslash \widehat{N}$.

(b) Let $\widehat{N}^k$ denote the admissible nodes set obtained in the beginning of iteration k. Then $\widehat{N}^k \subseteq \widehat{N}^{k+1}$ and $|\widehat{N}^{k+1}| \geq |\widehat{N}^k| + 1$.

(c) In at most $n-1$ major iterations, the algorithm *LSPD-1-1* terminates when node t becomes admissible. Then, s can send its unit imbalance to t via some path composed only by admissible arcs so that the total imbalance over all nodes becomes 0.

Thus finally when $\sum_{i\in N\backslash t} \delta_i^{*2}$ vanishes, which means the primal feasibility has been attained, and since the dual feasibility and complementary slackness conditions are maintained during the whole procedure, LSPD-1-1 solves the 1-1 shortest path problem from s to t. □



Iteration 1: Dual: $\pi = [0,0,0,0,0]$ $\widehat{A} = \emptyset$, $\widehat{N} = \{1\}$

RPP: $min\ 1^2$

$\delta^* = [1,0,0,0,0]$

$\widetilde{A} = \{(1,2),(1,3)\}$ $\theta = min\{2,1\} = 1$

Iteration 2: Dual: $\pi = [1,0,0,0,0]$ $\widehat{A} = \{((1,3)\}$, $\widehat{N} = \{1,3\}$

RPP: $min\ (1-x_{13})^2 + x_{13}^2$

$s.t.$ $x_{13} \geq 0$

$\delta^* = [\frac{1}{2},0,\frac{1}{2},0,0]$, $x_{13}^* = \frac{1}{2}$

$\widetilde{A} = \{(1,2),(3,5)\}$ $\theta = min\{2,2\} = 2$

Iteration 3: Dual: $\pi = [2,0,1,0,0]$ $\widehat{A} = \{((1,3),(1,2),(3,5)\}$, $\widehat{N} = \{1,3,2,5\}$

RPP: $min\ (1-x_{13}-x_{12})^2 + (x_{13}-x_{35})^2 + x_{12}^2 + x_{35}^2$

$s.t.$ $x_{13}, x_{12}, x_{35} \geq 0$

$\delta^* = [\frac{1}{4},\frac{1}{4},\frac{1}{4},0,\frac{1}{4}]$, $x_{13}^* = \frac{1}{2}, x_{12}^* = x_{35}^* = \frac{1}{4}$

$\widetilde{A} = \{(2,4),(5,4),(5,6)\}$ $\theta = min\{12,8,20\} = 8$

Iteration 4: Dual: $\pi = [4,2,3,0,2]$ $\widehat{A} = \{((1,3),(1,2),(3,5),(5,4)\}$, $\widehat{N} = \{1,3,2,5,4\}$

RPP: $min\ (1-x_{13}-x_{12})^2 + (x_{13}-x_{35})^2 + x_{12}^2 + (x_{35}-x_{54})^2 + x_{54}^2$

$s.t.$ $x_{13}, x_{12}, x_{35}, x_{54} \geq 0$

$\delta^* = [\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5}]$, $x_{13}^* = \frac{3}{5}, x_{35}^* = \frac{2}{5}, x_{12}^* = x_{54}^* = \frac{1}{5}$

$\widetilde{A} = \{(4,6),(5,6)\}$ $\theta = min\{10,15\} = 10$

Iteration 5: Dual: $\pi = [6,4,5,2,4]$ $\widehat{A} = \{((1,3),(1,2),(3,5),(5,4),(4,6)\}$, $\widehat{N} = \{1,3,2,5,4,6\}$

$t = 6 \in \widehat{N}$, Algorithm *LSPD-1-1* terminated.

Fig. 2. A small 1-1 shortest path example solved by algorithm *LSPD-1-1*.

Figure 2 illustrates how algorithm *LSPD-1-1* solves a 1-1 shortest path problem. Intuitively, we can view this algorithm as the following: starting from source s, *LSPD-1-1* tries to reach t by growing the set of admissible nodes. The algorithm keeps propagating the unit imbalance along all the admissible arcs so that the unit imbalance will be equally distributed to each admissible node before t becomes admissible. Once t becomes admissible, all imbalance flows to t so that the optimal system imbalance δ^* becomes 0. Then the algorithm is finished.

To further speed up algorithm *LSPD-1-1*, we observe that for each admissible node k, $\delta_k^* = \frac{1}{|\widehat{N}|}$ and $\theta\delta_k^* = \min_{(i,j)\in A, \delta_i^* > \delta_j^*}\{\frac{c_{ij}-\pi_i+\pi_j}{\delta_i^*-\delta_j^*}\} \cdot \frac{1}{|\widehat{N}|} = \min_{(i,j)\in A, \delta_i^* = \frac{1}{|\widehat{N}|}, \delta_j^* = 0}\{\frac{c_{ij}-\pi_i+\pi_j}{\frac{1}{|\widehat{N}|}}\} \cdot \frac{1}{|\widehat{N}|} = \min_{(i,j)\in A, \delta_i^* > \delta_j^*}\{c_{ij}-\pi_i+\pi_j\} \cdot 1$. Thus we can use $\widehat{\theta} = \min_{(i,j)\in A, \delta_i^* > \delta_j^*}\{c_{ij}-\pi_i+\pi_j\}$ and $\widehat{\delta}_k = 1$ to update $\pi_k = \pi_k + \widehat{\theta}\widehat{\delta}_k$ since $\theta\delta_k^* = \widehat{\theta}\widehat{\delta}_k$. This modification achieves the same objective using simpler computations.

When the original PD algorithm solves the 1-1 shortest path problem, the primal RPP formulation is as follows:

$$\min \sum_{i \in N \backslash t} \delta_i = \sum_{i \in N \backslash t} (b_i - \sum_{a \in \widehat{A}} \widehat{E}_{ia} x_a) \qquad \text{(RPP-1-1)}$$

$$\text{s.t.} \quad b_i = \begin{cases} 1, & \text{if } i = s \\ 0, & \text{if } i \in N \backslash \{s, t\} \end{cases}, \quad \forall\, i \in N \backslash t$$

$$x_a \geq 0 \quad \forall\, a \in \widehat{A},\ \delta_i \geq 0 \quad \forall\, i \in N \backslash t,$$

whose dual is

$$\max \rho_s \qquad \text{(DRPP-1-1)}$$

$$\begin{aligned} \text{s.t.} \quad & \rho_i \leq \rho_j, \quad \forall\, (i,j) \in \widehat{A}, i, j \neq t \\ & \rho_i, \leq 0, \quad \forall\, (i,t) \in \widehat{A} \\ & \rho_j, \geq 0, \quad \forall\, (t,j) \in \widehat{A} \\ & \rho_i, \leq 1, \quad \forall\, i \in N \backslash t. \end{aligned}$$

Unlike when solving the ALL-1 shortest path problem, the original PD algorithm will have degenerate pivots when solving RPP-1-1, which is a major difference from the LSPD algorithm since the LSPD algorithm guarantees nondegenerate pivots at every iteration.

If s and t are not adjacent and all the arc costs are strictly positive, we start the algorithm with $\pi = 0$ which makes $\widehat{A}$ empty in the first iteration. Then the optimal solution for DRPP-1-1 in the first iteration will be $\rho_s^* = 1,\ \rho_i^* \leq 1\ \forall\, i \in N \backslash \{s, t\}$. That is, we are free to choose any ρ_i^* as long as it does not exceed 1. This property of multiple optimal dual solutions is due to the degeneracy of RPP-1-1. When we have multiple choices to improve the dual solution π, there is no guarantee of improving the objective of RPP-1-1 at any iteration. In fact, we may end up cycling or take a long time to move out the degenerate primal solution.

To eliminate the uncertainty caused by primal degeneracy when solving RPP-1-1, we have to choose the dual improving direction appropriately. One way is to choose $\rho_i^* = 0$ for nonadmissible nodes. Then, by the first constraint in DRPP-1-1, admissible nodes will be forced to have $\rho_i^* = 1$. This is because we want to maximize ρ_s, and the best we can do is $\rho_s^* = 1$. By doing so, we force all the nodes reachable from s (i.e. admissible nodes) to have $\rho_i^* = 1$. Thus if the original PD algorithm chooses $\rho_i^* = 0$ for each nonadmissible node and $\rho_i^* = 1$ for each admissible node, then it will have chosen the same admissible arcs and nodes as *LSPD-1-1* in each iteration.

Dijkstra's algorithm for the 1-1 shortest path problem is the same as the ALL-1 case, except that it terminates as soon as the sink t is permanently labeled. Here we explain that *LSPD-1-1* performs the same operations as Dijkstra's algorithm does.

Algorithm *LSPD-1-1* starts at source node s, and then identifies admissible arcs to grow the set of admissible nodes. This is the same as Dijkstra's algorithm. If both algorithms choose the same node in each iteration, the admissible node set $\widehat{N}$ in the *LSPD-1-1* algorithm will be equivalent to the permanently labeled node set V in Dijkstra's algorithm.

The following proposition explains that both algorithms choose the same nodes in every major iteration.

Theorem 3.2. *Both Dijkstra and LSPD-1-1 choose the same node to become permanently labeled* (*in Dijkstra*) *or admissible* (*in LSPD-1-1*) *in each major iteration.*

Proof. See Appendix. □

Thus in *LSPD-1-1*, π_i represents the shortest distance between an admissible node i and the most recent admissible node in V^k, while in Dijkstra's algorithm $d(i)$ represents the shortest distance between s and i. In other words, $d(i) = \pi_s - \pi_i$ for any admissible node i. Therefore, when t becomes admissible, $d(t) = \pi_s$.

From this discussion, we can see that when solving the 1-1 shortest path problem with nonnegative arc lengths, Dijkstra and *LSPD-1-1* algorithms are, in fact, identical to each other. The original PD algorithm will face the problem of primal degeneracy when solving the RPP-1-1. However, if we choose the improving dual direction intelligently (i.e. $\rho^* = 0$ for all nonadmissible nodes and $\rho^* = 1$ for all admissible nodes), the original PD algorithm will perform the same operations as the *LSPD-1-1* algorithm. Next section compares the computational performance of *LSPD-1-1* and *PD-1-1* algorithms, and shows the effect of degeneracy.

4. Preliminary Computational Experiments

We have shown that all three algorithms, Dijkstra, *LSPD-ALL-1*, and PD, perform the same steps in solving ALL-1-SP problems, while the algorithm *PD-1-1* may perform different steps due to degeneracy when compared with Dijkstra and *LSPD-1-1* in solving 1-1-SP. To see how the degenerate pivots may affect the performance of *PD-1-1*, here we conduct several computational experiments for algorithms *LSPD-1-1* and *PD-1-1* in solving 1-1-SP on two families of artificial random networks, SPGRID and SPRAND, written by Cherkassky *et al.* (1996). SPGRID generates grid-like networks with X × Y grid nodes plus a super node. By changing X and Y we can specify the grid shape to be square (X = Y), wide or long (X ≠ Y). We specify the degree to be 3 and arc lengths to be ranged from 0 to 100, and generate four families of random grid networks where X and Y equal to 16 or 32. On the other hand, SPRAND first constructs a Hamiltonian cycle, and then adds arcs with distinct random end points. Here we set the length of the arcs to be uniformly chosen from the interval $[0, 10^4]$. All the algorithms and network generators are C codes compiled by GNU C compiler on an Intel Pentium 4 machine with 3.20 GHz CPU and 1 GB RAM.

Table 1. Comparison of algorithms *LSPD-1-1* and *PD-1-1* on SPGRID random networks.

Grid	[deg]	Time (ms)		P-D Iterations	
		LSPD-1-1	*PD-1-1*	*LSPD-1-1*	*PD-1-1*
16×16	[3]	2.37	9.30	234	379
16×32	[3]	8.19	32.51	401	688
32×16	[3]	6.32	34.06	302	706
32×32	[3]	22.18	114.28	542	1228

Table 2. Comparison of algorithms *LSPD-1-1* and *PD-1-1* on SPRAND random networks.

$\|N\|$	[deg]	Time (ms)		P-D Iterations	
		LSPD-1-1	*PD-1-1*	*LSPD-1-1*	*PD-1-1*
256	[4]	2.11	6.41	126	195
256	[16]	6.38	19.64	112	180
512	[4]	9.71	27.45	278	415
512	[16]	30.34	86.40	245	387
1024	[4]	31.88	99.41	458	756
1024	[16]	186.30	574.29	444	760

In order to catch the effect of degenerate pivots, we implement *PD-1-1* in a way that degenerate pivots may appear more often. In particular, instead of setting $\rho^* = 0$ for all nonadmissible nodes as a nondegenerate pivot, we set such ρ^* to be nonzero random numbers feasible for DRPP-1-1. Our implementation will still improve the dual solutions of 1-1-Dual, but is certainly different from the nondegenerate pivots.

We generate 10 random networks for each network family, and solve several 1-1-SP problems for each random network by choosing different origins and destinations. The average running time as well as the number of primal-dual iterations by both algorithms are recorded for comparison. Both Tables 1 and 2 show that the degenerate implementation of *PD-1-1* does spend more running time and conduct more primal-dual iterations to converge to the optimal solution, which verifies our expectation on the advantage of nondegenerate pivots performed by *LSPD-1-1*.

Note that here we give a specialized degenerate *PD-1-1* implementation for the purpose to observe the effect of degenerate pivots. In practice, popular LP solvers such as CPLEX or LINDO may give nondegenerate pivots (i.e. $\rho^* = 0$ for all nonadmissible nodes and $\rho^* = 1$ for all admissible nodes) when solving RPP-1-1, due to its simple mathematical structure.

5. Conclusions

The LSPD algorithm is more efficient than the original PD algorithm in the sense that it improves dual solutions in a nondegenerate way. When solving min-cost network flow problems, specialized LSPD implementation can avoid matrix inversion and performs efficient computation. Here we propose a more simplified LSPD

implementation which is shown to have identical steps to Dijkstra's algorithm for solving both ALL-1 and 1-1 shortest path problems. The original PD algorithm, on the other hand, is identical to the Dijkstra's algorithm for solving the ALL-1 shortest path problem, but only when we choose a specific dual improving direction (there are multiple ones) so that it will be identical to the Dijkstra's algorithm. The effect of degenerate pivots may slow down the performance of the original PD algorithm, as illustrated in our preliminary computational experiments, when compared with LSPD algorithm in solving the 1-1 shortest path problems. Since Dijkstra's algorithm is considered to be one of the most efficient algorithms in solving shortest path problems, this paper shows the potential of the LSPD algorithm and also provides more insights into the difference between the LSPD and the original PD algorithms.

Acknowledgments

I.-Lin Wang was partially supported by the National Science Council of Taiwan under Grant NSC93-2213-E006-096.

References

Ahuja, R, T Magnanti and J Orlin (1993). *Network Flows: Theory, Algorithms and Applications*. Englewood Cliffs, NJ: Prentice Hall.

Barnes, E, V Chen, B Gopalakrishnan and E Johnson (2002). A least-squares primal-dual algorithm for solving linear programming problems. *Operations Research Letters*, 30(5), 289–294.

Barnes, E, B Gopalakrishnan, E Johnson and J Sokol (2005). A least-squares network flow algorithm. in prepartion.

Cherkassky, B, A Goldberg and T Radzik (1996). Shortest paths algorithms: theory and experimental evaluation. *Mathematical Programming*, 73(2), 129–174.

Dijkstra, E (1959). A note on two problems in connection with graphs. *Numerische Mathematik*, 1, 269–271.

Gopalakrishnan, B, (2002). *Least-Squares Methods in Linear Programming*. Ph.D. thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Leichner, S, G Dantzig and J Davis (1993). A strictly improving linear programming phase i algorithm. *Annals of Operations Research*, 46–47(1–4), 409–430.

Padadimitriou, C and K Steiglitz (1982). *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ, Prentice-Hall.

Appendix

Proof of Theorem 2.2.. We already know that both algorithms start at the same node t. In *LSPD-ALL-1*, we will identify an admissible node j_1 by identifying the admissible arc (j_1, t) such that $(j_1, t) = \arg\min_{(j,t)\in A, \delta_j^* > \delta_t^*} \{\frac{c_{jt} - \pi_i + \pi_j}{\delta_j^* - \delta_t^*}\} = \arg\min_{(j,t)\in A}\{c_{jt}\}$. The second equality holds because $\delta_j^* = 1$, $\delta_t^* = 0$, and $\pi = 0$ in the first iteration. This is the same as Dijkstra's algorithm in the first iteration.

Assume both algorithms have the same set of admissible (or permanently labeled) nodes V^k in the beginning of the kth major iteration. Algorithm *LSPD-ALL-1* will choose an admissible arc (i_k, j_r) such that $(i_k, j_r) =$

$\arg\min_{(i,j)\in A,\delta_i^*>\delta_j^*}\{\frac{c_{ij}-\pi_i+\pi_j}{\delta_i^*-\delta_j^*}\} = \arg\min_{(i,j)\in A,j\in V^k,i\notin V^k}\{c_{ij}+\pi_j\}$. Again, the second equality holds because $\delta_j^* = 0$ for each $j \in V^k$, and $\delta_i^* = 1$, $\pi_i = 0$ for each $i \notin V^k$. If node j is admissible in the kth iteration, let $j \to j_h \to \cdots \to j_2 \to j_1 \to t$ denote the path from j to t. Then we can calculate $\pi_j = c_{jj_h} + \cdots + c_{j_2j_1} + c_{j_1t}$ since $\pi_t = 0$ and all the arcs along this path are admissible thus having zero reduced cost. So, $(i_k, j_r) = \arg\min_{(i,j)\in A,j\in V^k,i\notin V^k}\{c_{ij} + \pi_j\} = \arg\min_{(i,j)\in A,j\in V^k,i\notin V^k}\{\sum_{(p,q)\in path\{i\to j\to j_h\to\cdots\to j_1\to t\}} c_{pq}\}$. Therefore, in the beginning of the $(k+1)$st major iteration, node i_k becomes admissible with $\pi_{i_k} = \sum_{(p,q)\in path\{i_k\to j_r\to j_{r-1}\to\cdots\to j_1\to t\}} c_{pq}$.

Dijkstra's algorithm in the kth iteration will choose the node reachable from V^k with the minimum distance label. That is, one should choose node i_k reachable from a permanent labeled node j_r such that $d(i_k) = \min_{(j,i)\in A,j\in V^k} d(i) = \min_{(j,i)\in A,j\in V^k}\{d(j) + c_{ji}\}$. Let $t \to j_1 \to j_2 \to \cdots \to j_p \to j$ denote the path from t to a permanently labeled node j. Since j is permanently labeled, $d(j) = \sum_{(p,q)\in path\{j\to j_p\to\cdots\to j_2\to j_1\to t\}} c_{pq}$. Therefore, node i_k will become permanently labeled with distance label $d(i_k) = \sum_{(p,q)\in path\{t\to j_1\to j_2\to\cdots\to j_r\to i_k\}} c_{pq}$ in the $(k+1)$st major iteration.

Therefore, these two algorithms perform the same operation to get the same shortest distance label for the newly permanently labeled (or admissible) node. □

Proof of Theorem 3.2 We already know that both algorithms start at s. In *LSPD-1-1*, we will identify an admissible node i_1 by identifying the admissible arc (s, i_1) such that $(s, i_1) = \arg\min_{(s,i)\in A,\delta_s^*>\delta_i^*}\{\frac{c_{si}-\pi_s+\pi_i}{\delta_s^*-\delta_i^*}\} = \arg\min_{(s,i)\in A}\{c_{si}\}$. The second equality holds because $\delta_s^* = 1$, $\delta_i^* = 0$, and $\pi = 0$ in the first iteration. This is the same as Dijkstra's algorithm in the first iteration.

Assume both algorithms have the same set of admissible (or permanently labeled) nodes V^k in the beginning of the kth major iteration. Algorithm *LSPD-1-1* will choose an arc (i_r, j_k) such that $(i_r, j_k) = \arg\min_{(i,j)\in A,\delta_i^*>\delta_j^*}\{\frac{c_{ij}-\pi_i+\pi_j}{\delta_i^*-\delta_j^*}\} = \arg\min_{(i,j)\in A,i\in V^k,j\notin V^k}\{c_{ij} - \pi_i\}$. Again, the second equality holds because $\delta_i^* = 1$ for each $i \in V^k$, and $\delta_j^* = 0$, $\pi_j = 0$ for each $j \notin V^k$ If node i is admissible in the k^{th} iteration, let $s \to i_1 \to i_2 \to \cdots \to i_p \to i$ denote the path from s to i. Then we can calculate $\pi_s = c_{si_1} + c_{i_1i_2} + \cdots + c_{i_pi} + \pi_i$ since all the arcs along this path are admissible and thus have zero reduced cost. That is, $-\pi_i = \sum_{(p,q)\in path\{s\to i_1\to i_2\cdots\to i\}} c_{pq} - \pi_s$ for each admissible node i. So, $(i_r, j_k) = \arg\min_{(i,j)\in A,i\in V^k,j\notin V^k}\{c_{ij} - \pi_i\} = \arg\min_{(i,j)\in A,i\in V^k,j\notin V^k}\{\sum_{(p,q)\in path\{s\to i_1\to i_2\cdots\to i\to j\}} c_{pq} - \pi_s\} = \arg\min_{(i,j)\in A,i\in V^k,j\notin V^k}\{\sum_{(p,q)\in path\{s\to i_1\to i_2\cdots\to i\to j\}} c_{pq}\}$. The last equality holds since π_s is fixed when we compare all the arcs $(i, j) \in A, i \in V^k, j \notin V^k$.

Therefore, in the beginning of the $(k+1)$st major iteration, node j_k becomes admissible with $\pi_{j_k} = 0$, and $\pi_s = \sum_{(p,q)\in path\{s\to i_1\to i_2\cdots\to i_r\to j_k\}} c_{pq}$.

Dijkstra's algorithm in the kth iteration will choose a node reachable from V^k with a minimum distance label. That is, it chooses a node j_k reachable

from some permanent node i_r such that $d(j_k) = \min_{(i,j)\in A, i\in V^k, j\notin V^k} d(j) = \min_{(i,j)\in A, i\in V^k, j\notin V^k}\{d(i) + c_{ij}\}$. Let $s \to i_1 \to i_2 \to \cdots \to i_p \to i$ denote the path from s to a permanently labeled node i. Since i is permanently labeled, $d(i) = \sum_{(p,q)\in path\{s\to i_1\to i_2\cdots\to i_p\to i\}} c_{pq}$. Therefore, node j_k will become permanently labeled because $d(j_k) = \min_{(i,j)\in A, i\in V^k, j\notin V^k}\{d(i) + c_{ij}\} = \min_{(i,j)\in A, i\in V^k, j\notin V^k}\{\sum_{(p,q)\in path\{s\to i_1\to i_2\cdots\to i\to j\}} c_{pq}\}$. That is, node j_k will become permanently labeled in the $(k+1)$st major iteration and will have distance label $d(j_k) = \sum_{(p,q)\in path\{s\to i_1\to i_2\cdots\to i_r\to j_k\}} c_{pq}$.

It is easy to see that these two algorithms perform the same operation to identify the same newly permanently labeled (or admissible) node. □

I.-Lin Wang is an Associate Professor in the Department of Industrial & Information Management, National Cheng Kung University (IIM, NCKU), Tainan, Taiwan (since 2003). From 1996 to 1997, He served as a Foreign Researcher, in the Communication Network System Research Lab, Fujitsu Lab. Ltd, Kawasaki, Japan. He received Bachelor degree from the Department of Aeronautical & Astronautical Engineering, National Cheng Kung University (IAA, NCKU), Tainan, Taiwan in 1991 and Master of Science from the Operations Research Center, MIT (ORC, MIT), Cambridge, USA in 1996, and also Ph.D. from the School of Industrial & Systems Engineering, Georgia Institute of Technology (ISyE, GA Tech), Atlanta, USA in 2003. His research interests include, Network Optimization, or related topics, Logistics Supply Chain Management, and Bioinformatics. He has published articles in journals such as *Transportation Science*, *Journal of Industrial and Management Optimization*, *IEEE Transactions on Electronics Packaging Manufacturing* and *International Journal of Integrated Supply Management.*