Apply NNLS for Shortest Path Problem

I-Lin Wang

NNLS algorithm is an algorithm exploiting the original primal-dual(P-D) algorithm framework. Instead of using the linear objective to minimize the sum of slackness when solving the restricted primal problem(RP) as the original P-D algorithm does, NNLS tries to minimize the sum of square of the slackness.

Shortest path problem is a problem been researched for decades. The problem is to find shortest ways traveling between certain nodes in a graph $G(V,A)$ with $n = |V|$ nodes, $m = |A|$ arcs. Here we first evaluate how good the NNLS algorithm can be applied for graph whose arc length is always nonnegative.

1.0 NNLS for Shortest Path with Nonnegative arc length

Many combinatorial algorithms have been developed during the last 40yrs for solving the shortest path problem. For graphs with nonnegative arc length, Dijkstra's algorithm[1] has excellent running time in both theory and practice. We will explain that Dijkstra's algorithm is, in fact, a variant of the original P-D algorithm. We will also show that when applied to solve the shortest path problem with nonnegative arc length, NNLS algorithm is, in fact, identical to the Dijkstra's algorithm. First we discuss the All-1 shortest path problem, which has the special property that both the original P-D and NNLS algorithms will have nondegenerate pivots in every iteration.

1.1 All-1 Shortest Path Problem

The All-1 shortest path problem is to find a shortest path tree(SPT) rooted at a single sink node, *t*, reached by all the other nodes in the graph with shortest paths. We can consider such problem as having $(n-1)$ demands for node *t*, and we want to send 1 unit from every other node in the graph, in a way minimizing the total cost. When writing in mathematical programming format we get

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} \\ s.t. \quad & \\ & Nx = b \quad x \geq 0 \end{aligned}$$

N is the node-arc incidence matrix with entry $1, 0$ or -1. Column N_k corresponds to an arc (i,j) with i^{th} entry 1, j^{th} entry -1, and the others 0. b is the demand vector, b_i corresponds to demand of node i with entry $-(n-1)$ for sink node *t*, and 1 for all others. Let us reorder the matrix such that the last row corresponds to sink node *t*.

It can be shown that any node-arc incidence matrix for a connected graph has 1 redundant row. Therefore, we can just eliminate the last row for node *t*. Then the new incidence matrix $\bar{N}$ and demand vector $\bar{b}$ can be determined in the following way:

If arc $(i,t) \in A$, its corresponding column will only have one nonzero entry 1 in its i^{th} row. If

arc $(t,j) \in A$, its corresponding column will only have one nonzero entry –1 in its j^{th} row. Otherwise, the other columns just have the same nonzero row position as N. The demand vector b has 1 for all the $(n-1)$ entries.

The problem has the following primal and dual formulation:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \quad \text{(P)} \qquad \max \sum_{i\in N\setminus t} \pi_i \quad \text{(D)}$$

s.t. *s.t.*

$$\bar{N}x = \bar{b} \quad x \geq 0$$

$$\pi_i - \pi_j \leq c_{ij} \quad \forall (i,j) \in A, i,j \neq t \quad \pi \text{ free}$$
$$\pi_i \leq c_{it} \quad \forall (i,t) \in A$$
$$-\pi_j \leq c_{tj} \quad \forall (t,j) \in A$$

1.1.1 NNLS algorithm for ALL-1 shortest path problem

Given an initial feasible dual solution π (we can use $\pi = 0$ because $c_{ij} \geq 0$) for (D), first we identify those arcs $(i,j) \in A$ satisfying $\pi_i - \pi_j = c_{ij}$ as *admissible arcs*. Let $\bar{G}(V,\bar{E})$ denote the *admissible graph* which contains all the node set V but only admissible arc set $\bar{E}$ in G. We denote a node i as an *admissible node* if there exists a path from i to t in $\bar{G}$. Let the set of all admissible nodes be $\bar{V}$. We define $c_{ij}^{\pi} = c_{ij} - \pi_i + \pi_j$ as *reduced cost* of arc (i,j). We can construct the restricted primal(RP) problem by sending flow only via these admissible arcs, while minimizing the sum of squares of the node imbalance.

$$\min \sum_{i\in N\setminus t} s_i^2 \quad \text{(RP1)} \quad \text{, in other words,} \quad \min \sum_{i\in N\setminus t} (\bar{b}_i - \bar{E}_i x)^2 \quad \text{(RP2)}$$

s.t. *s.t.*

$$\bar{E}x + s = \bar{b} \quad x,s \geq 0 \qquad x \geq 0$$

$\bar{E}$ is a column subset of $\bar{N}$ corresponding to the admissible arcs. s is the node imbalance, or slackness vector. (RP2) is the so-called NNLS problem, which can be solved by the algorithm developed by Leichner and Dantzig [3]. Balaji et al [4] extend this algorithm to solve general LP problems. They use the framework of the original Primal-Dual algorithm which is a dual-ascent method maintaining dual feasibility and complementary slackness condition while trying to achieve primal feasibility along an improving dual feasible direction obtained by solving (RP). Details about why the optimal slackness of (RP1) can be used as a dual improving direction can be found in the paper by Balaji. et al [4]. The algorithm NNLS used here is a much simpler version than its original one in [4] by exploiting the special structure of ALL-1 shortest path problem.

The primal-dual process is as follows:

Algorithm P-D-NNLS-ALL-1(G)

Step 1: Set initial dual feasible solution $\pi_i = 0$, for each node i; let $s_t = 0$

Step 2: Identify admissible arc set $\bar{E}$; if $|E| = n-1$, then **STOP**

Step 3: Apply **Algorithm NNLS-ALL-1**($\bar{G}, \bar{V}, s$) , get slack vector s

Step 4: Update dual feasible solution $\pi = \pi + \theta \bullet s$, $\theta = \min_{(i,j)\in A, s_i > s_j} \{\frac{c_{ij} - \pi_i + \pi_j}{s_i - s_j}\}$;

GOTO Step 2

Algorithm NNLS-ALL-1($\bar{G}, \bar{V}, s$)
Step 1: Identify set of admissible nodes $\bar{V}$
Step 2: Compute slackness for each node $i \in N$:
if node $i \in \bar{V}$, then $s_i = 0$
else $s_i = 1$

Applying the algorithm P-D-NNLS-ALL-1, we get the following 3 observation:
1. An admissible node can always send flow to t so that it will have zero slackness, otherwise its slackness remains 1.
2. Admissible nodes set in iteration k, $\bar{V}^k$, is a subset of the admissible nodes set in the next iteration, $\bar{V}^k$. That is, $\bar{V}^k \subseteq \bar{V}^{k+1}$.
3. Algorithm ends when $\bar{V} = V$

Why NNLS-ALL-1 can successfully compute the optimal slackness s_i^* for (RP1) will be stated in the following Lemma.

Lemma 1. When used in Algorithm P-D-NNLS-ALL-1, Algorithm NNLS-ALL-1($\bar{G}, \bar{V}, s$) will correctly compute the optimal slackness for (RP1)

Proof:
This is the result of the first observation above. To solve (RP1), we want to minimize $\sum_{i \in N \setminus t} s_i^2$ using only the admissible arcs $\bar{E}$. For those nodes that are not admissible, they can't send flow along admissible arcs in $\bar{G}$, thus their slackness remains the same as $b_i = 1$.

In the shortest path problem, all the arcs do not have the capacity constraints. Therefore, as long as node i is admissible, that is, there exists a path from i to t, and thus we can always send 1 unit of flow from i to t to make $s_i = 0$.

Therefore the algorithm NNLS-ALL-1 will correctly compute the optimal slackness for (RP1).

Next we will compare the original P-D algorithm which only used linear slack when solving (RP) with NNLS for ALL-ALL shortest path problem.

1.1.2 NNLS vs Original P-D algorithm for ALL-1 shortest path problem

These two algorithms only differ with each other in solving the Restricted Primal(RP) problem in the P-D process. The original P-D solves the following Restricted Primal problem:

$$\min \sum_{i \in N \setminus t} s_i \qquad \text{(RP3)}$$

s.t.

$$\bar{E}x + s = \bar{b} \quad x, s \geq 0$$

whose dual is

$$\max \sum_{i \in N \setminus t} \varrho_i \qquad \text{(DRP3)}$$

s.t.

$$\begin{aligned} \varrho_i - \varrho_j &\le 0 \quad \forall (i,j) \in \bar{E}, i, j \ne t \\ \varrho_i &\le 0 \quad \forall (i,t) \in \bar{E} \\ -\varrho_j &\le 0 \quad \forall (t,j) \in \bar{E} \\ \varrho_i &\le 1 \quad \forall i \in N \setminus t \end{aligned}$$

The optimal dual solution ϱ^* for (DRP3) will be used as a dual-ascent direction in the P-D process. It is easy to observe that for each node *i* that can't reach *t* along admissible arcs $\bar{E}$, we will always make $\varrho_i^* = 1$. Also, if node *i* is admissible, that is, there exists a path from *i* to *t* with intermediate nodes $\{i_1, i_2, i_3, ..., i_k\}$, then $\varrho_{i_1}^* = \varrho_{i_2}^* = \varrho_{i_3}^* = ... = \varrho_{i_k}^* = 0$. In other words, all the admissible nodes will have $\varrho^* = 0$, and $\varrho^* = 1$ for the remaining nodes that are not admissible. This improving direction is identical to the one of NNLS-ALL-1 in previous subsection.

Therefore we can say these two algorithms are, in fact, identical to each other. Next we will compare these two algorithms with the famous Dijkstra's algorithm.

1.1.3 NNLS vs Dijkstra's algorithm for ALL-1 shortest path problem

First let's review the Dijkstra's algorithm. For our convenience, we reverse all the arc direction so that solving the original ALL-1 shortest path problem is the same as solving the 1-ALL shortest problem with nonnegative arc length :

Algorithm Dijkstra(*G*)
begin
  $S := \emptyset; \bar{S} := N; d(i) := \infty \; \forall i \in N; d(t) := 0, pred(t) := 0;$
  while $|S| < n$ **do**
  begin
    let $i \in \bar{S}$ be a node with $d(i) = \min\{d(j) : j \in \bar{S}\}$
    $S := S \cup \{i\}; \bar{S} := \bar{S} - \{i\}$
    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
end

We say a node is *permanently labeled* if it is put in *S*. A node is *temporarily labeled* if it is reachable from any permanently labeled node. A node is labeled if it's either permanently labeled or temporarily labeled. Note that $d(i) := \infty$ if *i* is not labeled.

Dijkstra's algorithm starts to label *t* , then keep labeling nodes from the permanently labeled nodes. This is identical to the P-D-NNLS-ALL-1 which grows admissible nodes only from admissible nodes. In fact, in every major iteration, the set of admissible nodes in P-D-NNLS-ALL-1 is the same as the set of permanently labeled nods in Dijkstra. To show this, we only need to show that both algorithms

will choose the same nodes in every major iteration.

Lemma 2.Both algorithm Dijkstra and P-D-NNLS-ALL-1 choose the same node to become permanently labeled(in Dijkstra) or admissible(in P-D-NNLS-ALL-1) in each iteration.

Proof:

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

Assume both algorithms choose the same nodes $\bar{V}$ in the beginning of the k^{th} major iteration. Now in the Step 4 of the k^{th} iteration, the P-D-NNLS-ALL-1 will choose an admissible arc (i_k, j_r) such that $(i_k, j_r) = \arg \min_{(i,j)\in A, s_i > s_j} \{\frac{c_{ij} - \pi_i + \pi_j}{s_i - s_j}\} = \arg \min_{(i,j)\in A, j\in\bar{V}, i\notin\bar{V}} \{c_{ij} + \pi_j\}$. Again, the second equality holds because $s_j = 0$ for each $j \in \bar{V}$, and $s_i = 1$, $\pi_i = 0$ for each $i \notin \bar{V}$. If node *j* is admissible in the k^{th} iteration, let $j - j_p - ... - j_2 - j_1 - t$ denote the path from *j* to *t*. Then we can calculate $\pi_j = c_{jj_p} + ... + c_{j_2 j_1} + c_{j_1 t}$ since $\pi_t = 0$ and all the arcs along this path are admissible thus having zero reduced cost. So, $(i_k, j_r) = \arg \min_{(i,j)\in A, j\in\bar{V}, i\notin\bar{V}} \{c_{ij} + \pi_j\} = \arg \min_{(i,j)\in A, j\in\bar{V}, i\notin\bar{V}} \{\sum_{(p,q)\in path\{i-j-j_p-...-j_1-t\}} c_{pq}\}$. Therefore, in the beginning of the $(k+1)^{th}$ major iteration, node i_k becomes admissible with $\pi_{i_k} = \sum_{(p,q)\in path\{i_k - j_r - j_{r-1} - ... - j_1 - t\}} c_{pq}$.

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

It is easy to see that these two algorithms perform the same operation to get the same shortest distance label for the same newly permanently labeled(or admissible) node.

From these discussion, we conclude that when solving the ALL-1 shortest path problem with nonnegative arc length, all the three algorithms: Dijkstra, P-D-NNLS-ALL-1, and the original P-D , will perform the same operation in each iteration. In fact, this result is due to the nondegeneracy of the problem structure. Remember that the basis corresponding to a spanning tree in the network problem. In this ALL-1 shortest path problem, each node has supply 1, therefore the spanning tree arc will always have positive flow, which means the pivot will always be nondegenerate.

P-D–NNLS-ALL-1 is an algorithm designed to take advantage of doing nondegenerate pivots in each iteration. Therefore, in this special case, it just does as well as others. However, later we will see that the 1-1 shortest path problem does not have such nondegenerate property any more, thus P-D-NNLS-1-1 does do better job than the original P-D algorithm in some way.

1.2 1-1 Shortest Path Problem

Unlike the ALL-1 shortest path problem which we will try to find the shortest path tree(SPT), here we only need part of the SPT, namely, the shortest path between certain two node, *s* and *t* in a graph $G(V,A)$. We can think it as sending a unit flow from *s* to *t* with the minimal cost. Its mathematical programming format is the same as the (P) in section 1.1 except now the node imbalance vector *b* only has two nonzero entries: +1 for *s*, –1 for *t*, and 0 for all the other nodes.

Since the node-arc incidence matrix has one redundant row, we remove the one corresponding to *t*, then we get the following primal and dual formulation:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \qquad \text{(P')}$$

$$s.t. \quad \bar{N}x = \bar{b} \quad x \geq 0$$

$$\max \ \pi_s \qquad \text{(D')}$$

$$s.t.$$

$$\pi_i - \pi_j \leq c_{ij} \quad \forall (i,j) \in A, i,j \neq t \quad \pi \text{ free}$$

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

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

Here $\bar{b}$ only has one nonzero entry: +1 for *s*. This makes the difference between (D') and (D) in section 1.1 in that (D') only maximizes π_s. We have to modify the NNLS algorithm for this change of problem structure. We will also illustrate the impact of this difference when we apply the original P-D algorithm to solve this 1-1 shortest path problem. Finally we will explain the relation between these three algorithms when used to solve this specific problem.

1.2.1 NNLS algorithm for 1-1 shortest path problem

We will need to redefine the *admissible node* set $\bar{V}$ as follows: a node *i* is admissible if it is reachable from *s* only via admissible arcs. All the other definition such as admissible arcs and admissible graph remains the same as in section 1.1. When writing in mathematical format, we get:

$$\min \sum_{i\in N\setminus t} s_i^2 \qquad \text{(RP'1)}$$

$$s.t. \quad \bar{E}x + s = \bar{b} \quad x,s \geq 0$$

, in other words,

$$\min \sum_{i\in N\setminus t} (\bar{b}_i - \bar{E}_i x)^2 \qquad \text{(RP'2)}$$

$$s.t. \quad x \geq 0$$

The restricted primal (RP'1) problem has the same formulation as (RP1) in subsection 1.1.1 except $\bar{b}$ only has one nonzero entry: +1 for *s*. Therefore we modify the NNLS algorithm as follows:

Algorithm P-D-NNLS-1-1(G)
Step 1: Set initial dual feasible solution $\pi_i = 0$, for each node *i*; let $s_t = 0$
Step 2: Identify admissible arc set $\bar{E}$
Step 3: Apply **Algorithm NNLS-1-1**($\bar{G}, \bar{V}, s$) , get slack vector *s*;
Step 4: Update dual feasible solution $\pi = \pi + \theta \bullet s$, $\theta = \min_{(i,j)\in A, s_i > s_j} \{\frac{c_{ij} - \pi_i + \pi_j}{s_i - s_j}\}$;

GOTO Step 2

Algorithm NNLS-1-1($\overline{G}, \overline{V}, s$)
Step 1: Identify set of admissible nodes $\overline{V}$
if $t \in \overline{V}$, then **STOP**
Step 2: Compute slackness for each node $i \in N$:
if node $i \in \overline{V}$, then $s_i = \frac{1}{|\overline{V}|}$
else $s_i = 0$

Applying the algorithm P-D-NNLS-1-1, we get the following 3 observation:
1. Source s is the first admissible node identified. The unit node imbalance in s will equally distributed to all the admissible nodes in $\overline{V}$.
2. Admissible nodes set in iteration k, $\overline{V}^k$, is a subset of the admissible nodes set in the next iteration, $\overline{V}^k$. That is, $\overline{V}^k \subseteq \overline{V}^{k+1}$.
3. Algorithm ends when the sink t becomes admissible. Then s can send the unit flow to t via only some path composed of admissible arcs so that the slackness vector becomes 0.

Why NNLS-1-1 can successfully compute the optimal slackness s_i^* for (RP'1) will be stated in the following Lemma.

Lemma 3. When used in Algorithm P-D-NNLS-1-1, Algorithm NNLS-1-1($\overline{G}, \overline{V}, s$) will correctly compute the optimal slackness for (RP'1)

Proof:
This is the result of the first observation above. To solve (RP'1), we want to minimize $\sum_{i \in N \setminus t} s_i^2$ using only the admissible arcs $\overline{E}$. For those nodes that are not admissible, they can't receive or send flow along admissible arcs in $\overline{G}$, thus their slackness remains the same as $b_i = 0$. Therefore we only need to minimize $\sum_{i \in \overline{V}} s_i^2$.

For those admissible nodes, only $b_s = 1$, all the others has 0 imbalance. Since each admissible arc has no capacity constraint, for this special case, the optimal least square solution will always be nonnegative. Thus this unit imbalance will flow around $\overline{G}$ such that $\sum_{i \in \overline{V}} s_i^2$ is minimized while maintain the flow in all the admissible arcs to be nonnegative. [4] gives details about how to compute the optimal slack vector s^* and optimal flow vector x^* for (RP'1), which is identical to the step 2 in the algorithm NNLS-1-1.

Intuitively, we can view this algorithm as starting from s we try 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, suddenly all the imbalance flows to t so that slack vector s becomes 0. Thus we terminate the algorithm.

In fact, we can skip the computation of fractional s_i and θ by the following modifications:
1. In the Step 2 of NNLS-1-1, we make $\hat{s}_i = 1$ if node $i \in \overline{V}$
2. In the Step 4 of P-D-NNLS-1-1, we choose $\hat{\theta} = \min_{(i,j) \in A, s_i > s_j} \{c_{ij} - \pi_i + \pi_j\}$

If node k is not admissible, $\theta \bullet s_k = 0$. Otherwise, $s_k = \frac{1}{|\overline{V}|}$, and $\theta \bullet s_k = \min_{(i,j) \in A, s_i > s_j} \left\{\frac{c_{ij} - \pi_i + \pi_j}{s_i - s_j}\right\} \bullet \frac{1}{|\overline{V}|}$

$= \min_{(i,j)\in A, s_i=\frac{1}{|\bar{V}|}, s_j=0} \{\frac{c_{ij}-\pi_i+\pi_j}{\frac{1}{|\bar{V}|}}\} \bullet \frac{1}{|\bar{V}|} = \min_{(i,j)\in A, s_i>s_j} \{c_{ij} - \pi_i + \pi_j\} \bullet 1 = \hat{\theta} \bullet \hat{s}_k$. Therefore, making these modifications will prevent us computing the fractional overhead, while achieving the same result.

For our convenience, we will use this modified version as the P-D-NNLS-1-1 algorithm for the future discussion from now on.

1.2.2 NNLS vs Original P-D algorithm for 1-1 shortest path problem

Unlike the ALL-1 shortest path problem, the original P-D algorithm will have degenerate pivot in solving the restricted primal problem(RP'3), which makes the major difference with the NNLS algorithm.

$$\min \sum_{i\in N\setminus t} s_i \qquad \text{(RP'3)}$$

s.t.

$$\bar{E}x + s = \bar{b} \quad x, s \geq 0$$

whose dual is

$$\max \ \varrho_s \qquad \text{(DRP'3)}$$

s.t.

$$\begin{aligned} &\varrho_i - \varrho_j \leq 0 && \forall (i,j) \in \bar{E}, i, j \neq t \\ &\varrho_i \leq 0 && \forall (i,t) \in \bar{E} \\ &-\varrho_j \leq 0 && \forall (t,j) \in \bar{E} \\ &\varrho_i \leq 1 && \forall i \in N \setminus t \end{aligned}$$

If s and t are not adjacent, we start the algorithm with $\pi = 0$ which usually makes $\bar{E}$ empty in the first iteration if there is no arc with zero cost. Then the optimal solution for (DRP'3) in the first iteration will be $\varrho_s^* = 1, \varrho_i^* \leq 1 \ \forall i \in N \setminus \{s,t\}$. That is, we are free to choose any ϱ_i^* as long as it does not exceed 1. This property of multiple optimal dual solution is due to the degeneracy of (RP'3). When we have multiple choices to improve the dual solution, there is no guarantee to improve the objective of (RP'3) at any iteration. In fact, we may end up with cycling around or take a long time to move out the degenerate primal solution. If we are very lucky by choosing the "right" dual improving direction, we may even solve this problem much quicker.

To get rid of this uncertainty caused by the degeneracy when solving the restricted primal problem, we have to choose the dual improving direction smartly. One way is to choose $\varrho_i^* = 0$ for those nodes which are not admissible in $\bar{G}$. Then by the first constraint in (DRP'3), those admissible nodes will be forced to have $\varrho_i^* = 1$. This is because we want to maximize ϱ_s, and the best we can do is $\varrho_s^* = 1$. By doing so, we force all the nodes reachable from s (i.e. admissible nodes) to have $\varrho_i^* = 1$.

If we do avoid degeneracy by the method explained in the previous paragraph, actually we will get the same algorithm as the modified P-D-NNLS-1-1 algorithm stated in the end of subsection 1.2.1. Moreover, this modified algorithm is in fact, Dijkstra's algorithm.

1.2.3 NNLS vs Dijkstra's algorithm for 1-1 shortest path problem

The Dijkstra's algorithm for 1-1 shortest path problem is the same as the one in subsection 1.1.3, except that we will terminate the algorithm as soon as *t* is permanently labeled.

P-D-NNLS-1-1 starts at source node *s*, then identify 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 $\bar{V}$ in the P-D-NNLS-1-1 algorithm will be equivalent to the permanently labeled node set $\bar{S}$ in Dijkstra's algorithm. Lemma 4 explains why both algorithms will have the same operations in each iteration.

Lemma 4.Both algorithm Dijkstra and P-D-NNLS-1-1 choose the same node to become permanently labeled(in Dijkstra) or admissible(in P-D-NNLS-1-1) in each iteration.

Proof:

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

Assume both algorithms choose the same nodes $\bar{V}$ in the beginning of the k^{th} major iteration. Now in the Step 4 of the k^{th} iteration, the P-D-NNLS-1-1 will choose an arc (i_r, j_k) such that $(i_r, j_k) = \arg \min_{(i,j)\in A, s_i > s_j} \{\frac{c_{ij} - \pi_i + \pi_j}{s_i - s_j}\} = \arg \min_{(i,j)\in A, i\in \bar{V}, j\notin \bar{V}} \{c_{ij} - \pi_i\}$. Again, the second equality holds because $s_i = 1$ for each $i \in \bar{V}$, and $s_j = 0$, $\pi_j = 0$ for each $j \notin \bar{V}$ If node *i* is admissible in the k^{th} iteration, let $s - i_1 - i_2 - ... - i_p - i$ denote the path from *s* to *i*. Then we can calculate $\pi_s = c_{si_1} + c_{i_1 i_2} + ... + c_{i_p i} + \pi_i$ since all the arcs along this path are admissible thus having zero reduced cost. That is, $-\pi_i = \sum_{(p,q)\in path\{s - i_1 - i_2 ... - i\}} c_{pq} - \pi_s$ for each admissible node *i*. So, $(i_r, j_k) = \arg \min_{(i,j)\in A, i\in \bar{V}, j\notin \bar{V}} \{c_{ij} - \pi_i\} =$ $\arg \min_{(i,j)\in A, i\in \bar{V}, j\notin \bar{V}} \{\sum_{(p,q)\in path\{s - i_1 - i_2 ... - i - j\}} c_{pq} - \pi_s\} = \arg \min_{(i,j)\in A, i\in \bar{V}, j\notin \bar{V}} \{\sum_{(p,q)\in path\{s - i_1 - i_2 ... - i - j\}} c_{pq}\}$. The last equality holds since π_s is fixed when we compare all the arcs $(i, j) \in A, i \in \bar{V}, j \notin \bar{V}$.

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

The criterion to choose the new admissible arc (i_r, j_k) is the same as Dijkstra's algorithm which we will explain next.

The Dijkstra's algorithm in the k^{th} iteration will choose a node reachable from $\bar{V}$ with minimal distance label. That is, choose node j_k reachable from some permanent node i_r such that $d(j_k) = \min_{(i,j)\in A, i\in \bar{V}, j\notin \bar{V}} d(j) = \min_{(i,j)\in A, i\in \bar{V}, j\notin \bar{V}} \{d(i) + c_{ij}\}$. Let $s - i_1 - i_2 - ... - i_p - i$ denote the path from *s* to a permanently labeled node *i*. Since *i* is permanently labeled, $d(i)$

$= \sum_{(p,q)\in path\{s-i_1-i_2...-i_p-i\}} c_{pq}$. Therefore, node j_k will become permanently labeled because $d(j_k) = \min_{(i,j)\in A, i\in\overline{V}, j\notin\overline{V}} \{d(i) + c_{ij}\} = \min_{(i,j)\in A, i\in\overline{V}, j\notin\overline{V}} \{ \sum_{(p,q)\in path\{s-i_1-i_2...-i-j\}} c_{pq}\}$

Therefore, node j_k will become permanently labeled in the $(k+1)^{th}$ major iteration with distance label $d(j_k) = \sum_{(p,q)\in path\{s-i_1-i_2...-i_r-j_k\}} c_{pq}$.

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

From the proof of Lemma 4, we observe that in P-D-NNLS-1-1 algorithm, π_i represents the shortest distance between an admissible node *i* and the most recent admissible node in $\overline{V}$, 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, $d(t) = \pi_s$.

Here we use a physical example to illustrate these two algorithms. Consider each node as a ball, and each arc as a string connecting two balls. Here we assume all the arc length is positive.

For P-D-NNLS-1-1 algorithm, we can think it as follows: in the beginning we put all the balls on the floor, then we pick ball *s* to hang up. We keep hanging up *s* until the string (s, i_1) becomes tight, then ball i_1 will be hung up. We keep increasing the height of *s* until finally ball *t* is going to be hung up. At this moment, the height of *s* (i.e. π_s) is the shortest path between *s* and *t*.

For Dijkstra's algorithm, we use a plate which has one holes for these *n* balls to fall through. In the beginning we put the plate on the floor, then put ball *s* in the hole and all the other balls on the plate. We will stick ball *s* on the floor so that when we raise the plate, *s* will stay on the floor. Then we begin to raise the plate until the string (s, i_1) is so tight that ball i_1 begin the pass through the hole. We keep raising the plate until finally ball *t* is going to fall through the hole. At that time, the height of ball *t* (i.e. $d(t)$) is the shortest path between *s* and *t*.

From these discussion, we conclude that when solving the 1-1 shortest path problem with nonnegative arc length, Dijkstra and P-D-NNLS-1-1 algorithm are in fact, identical to each other. The original P-D algorithm will face the problem of degeneracy when solving the restricted primal problem. However, if we choose the improving dual direction smartly, the original P-D algorithm may have same operations as the P-D-NNLS-1 algorithm.

2.0 NNLS for Shortest Path with Nonnegative arc length

Reference

[1] E.W. Dijkstra, "A note on two problems in connexion with graphs". *Numeriche Mathematics*, vol.1,, pp. 269-71, 1959

[2] C.H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization*: *Algorithms and Complexity*, Englewood Cliffs, New Jersey : Prentice-Hall, 1982

[3] S.A. Leichner, G.B. Dantzig, J.W. Davis, "A strictly improving linear programming Phase I algorithm", *Annals of Operations Research*, vol.46-47, no.1-4, pp. 409-30, Dec. 1993

[4] Balaji