A COMBINATORIAL ARC TOLERANCE ANALYSIS FOR NETWORK FLOW PROBLEMS

P. T. SOKKALINGAM AND PRABHA SHARMA

Received 10 July 2002 and in revised form 15 May 2003

For the separable convex cost flow problem, we consider the problem of determining tolerance set for each arc cost function. For a given optimal flow x, a valid perturbation of $c_{ij}(x)$ is a convex function that can be substituted for $c_{ij}(x)$ in the total cost function without violating the optimality of x. Tolerance set for an arc(i,j) is the collection of all valid perturbations of $c_{ij}(x)$. We characterize the tolerance set for each arc(i,j) in terms of nonsingleton shortest distances between nodes i and j. We also give an efficient algorithm to compute the nonsingleton shortest distances between all pairs of nodes in $O(n^3)$ time where n denotes the number of nodes in the given graph.

1. Introduction

Consider a directed network $G = (N,A)$, where N is the set of nodes and A is the set of arcs. Let x denote the flow vector and let $C_{ij}(x)$ be the convex cost function associated with each arc$(i,j) \in A$, where $C_{ij}(x)$ is a function of the flow x_{ij} on arc(i,j) only. The separable convex cost flow (SCCF) problem is to find a flow x which minimizes the total cost while satisfying supplies/demands at nodes and honoring the capacity constraints on the arcs. The arc tolerance analysis for SCCF is the problem of characterizing valid cost perturbations of an arc with respect to an optimal flow; the valid perturbations are the convex cost functions that can replace the existing cost function of an arc while keeping the optimality of the given flow. In this paper, we study the arc tolerance analysis for SCCF and give a combinatorial algorithm to compute the arc tolerance set for each arc which characterizes the corresponding valid perturbations.

For the minimum cost flow problem, the arc cost function $C_{ij}(x) = c_{ij}x_{ij}$ is a linear function of the arc flow and c_{ij} is called the cost coefficient of arc(i,j). The set of all the values of c_{ij} for which a flow remains optimal forms an interval on the real line. In the literature, the problem of finding these intervals is called the arc tolerance analysis (Shier and Witzgall [7]) or the cost sensitivity analysis (Srinivasan and Thompson [9], Maier [4]). Rockafellar [6] outlines a method to compute the characteristic curve of the optimal value of the objective function for SCCF, when the arc under consideration has a linear cost function and the slope of this function is varied.

DOI: 10.1155/JAMDS.2005.83

Srinivasan and Thompson [9] specialized sensitivity results of linear programming to the transportation problem. They apply the so-called *basis preserving cell operators* to calculate tolerance intervals associated with an optimal basis. Shier and Witzgall [7] studied arc tolerance associated with a basis for the linear minimum cost flow problem. They exploited the tree structure of the basis to develop an $O(n^2)$ algorithm to calculate tolerance intervals for all arcs.

Optimal basic (spanning tree) solutions to network flow problems often have high degree of degeneracy. For example, dynamic networks for the vehicle allocation problem studied by Powell [5] have been found to have 40%–80% degenerate basic (tree) arcs (i.e., flows on these arcs are either at upper or lower bounds). Due to degeneracy, tolerance interval of an arc with respect to an optimal basis is often not exact, that is, it need not contain all possible perturbations of the arc cost coefficient. Degeneracy issues have also drawn the attention of other researchers. Fong and Srinivasan [2] calculate nondegenerate shadow prices. They delete the degenerate basic arcs, contract the resultant subtrees into a single node, and then find shortest distances in the resultant graph to find nondegenerate shadow prices for the transportation problem. Powell [5] has characterized nondegenerate shadow prices for the minimum cost flow problem in terms of shortest distances without using any contractions. He gives an approximation procedure for calculating nondegenerate shadow prices, and applies it to dynamic networks arising in the vehicle allocation problem.

Maier [4] has modified Fong and Srinivasan's procedure for calculating nondegenerate arc tolerance intervals for the minimum cost flow problem. He calculates the nondegenerate tolerance interval of a nondegenerate basic arc by deleting all degenerate basic arcs and the considered arc, and then contracting the resulting subtrees. The shortest distance between the head and tail nodes of the considered arc gives the required interval. For the SCCF, we define the arc tolerance set for an arc(k,l) with respect to an optimal flow (say $\bar{x}$), as the set having the property that any convex cost function with at least one of its right or left derivative or both at $\bar{x}$, in this set, is a valid cost perturbation of the arc(k,l). In this paper, we show that the arc tolerance set has one of the following forms: $\{(-\infty,\tilde{d}_k(l)],[-\tilde{d}_l(k),\infty)\}$, $[-\tilde{d}_l(k),\infty)$, or $(-\infty,-\tilde{d}_k(l)]$. Here $\tilde{d}_l(k)$ is the nonsingleton shortest distance from node l to node k and $\tilde{d}_k(l)$ is the nonsingleton shortest distance from node k to node l. We develop an $O(n^3)$ algorithm to compute the nonsingleton shortest distance $\{\tilde{d}_k(l)\}$ for all pairs of nodes. For the minimum cost flow problem, our arc tolerance set reduces to an exact tolerance interval, and our algorithm does not have to contract the original network for computing these intervals. The paper is organized as follows. Section 2 contains the necessary background material and the optimality conditions for the separable convex cost flow problem. In Section 3, we characterize the arc tolerance set in terms of nonsingleton shortest distances. In Section 4, we develop the algorithm for computing nonsingleton shortest distances between all pairs of nodes. Section 5 contains concluding remarks.

2. Notation and background

In this section, we state the separable convex cost flow (SCCF) problem and introduce related definitions and notations. We also state the optimality conditions for a flow to be optimal for SCCF. We follow the terminologies and notations used in [1] apart from those defined in this paper:

(i) $G = (N, A)$, a directed network;
(ii) N = node set, $|N| = n$;
(iii) A = arc set, $|A| = m$;
(iv) $x = \{x_{ij}\}$, m-dimensional flow vector on the arc set A;
(v) $C_{ij}(x)$ = convex cost function for flow x_{ij} on arc(i, j);
(vi) u_{ij} = upper bound on the flow on arc(i, j);
(vii) b_i = supply/demand for node $i \in N$, $\sum_{i=1}^{n} b(i) = 0$.

SCCF can be formulated as follows:

$$\text{Minimize } C(x) = \sum_{(i,j)\in A} C_{ij}(x) \tag{2.1}$$

subject to

$$\begin{gathered}\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \quad \forall i \in N, \\ 0 \le x_{ij} \le u_{ij} \quad \forall (i,j) \in A.\end{gathered} \tag{2.2}$$

The SCCF reduces to the minimum cost flow problem, when all $C_{ij}(x)$'s are linear functions; that is, $C_{ij}(x) = c_{ij}x_{ij}$, where c_{ij} is a real number.

A large variety of algorithms, which either use nonlinear programming techniques or linear programming techniques, are available for solving the SCCF. Ahuja et al. [1] have given quite a comprehensive list of references on SCCF.

2.1. Marginal costs. The right marginal cost of arc(k,l), $C_{kl}^{+}(x)$, with respect to the flow, is the right derivative of the arc cost function $C_{kl}(x)$ at $\overline{x}$, that is, the rate of change in $C_{kl}(\overline{x})$ when flow is increased on arc(k,l). Thus

$$C_{kl}^{+}(\overline{x}) = \lim_{\delta\to 0} \frac{(C_{kl}(\overline{x}+\delta e_{kl}) - C_{kl}(\overline{x}))}{\delta}, \quad \delta \ge 0, \tag{2.3}$$

where e_{kl} is an m-dimensional vector of zeros with a one in the component corresponding to the arc(k,l). Similarly, the left marginal cost of arc(k,l), $C_{kl}^{-}(\overline{x})$, is the left derivative of the arc cost function $C_{ij}(x)$ at $\overline{x}$; and is negative of the rate of change in $C_{ij}(\overline{x})$ when flow is decreased on arc(k,l), that is,

$$C_{kl}^{-}(\overline{x}) = \lim_{\delta\to 0} \frac{(C_{kl}(x-\delta e_{kl}) - C_{kl}(x))}{-\delta}, \quad \delta \ge 0. \tag{2.4}$$

2.2. Cost of augmenting cycles and paths. A cycle W in G is an augmenting cycle with respect to a flow $\overline{x}$ if we can augment $\theta > 0$ units of flow along the cycle while satisfying (2.2). We define the cost $C(W)$ of the cycle W as the rate of change in the objective function as we augment the flow along the cycle W. Thus,

$$C(W) = \lim_{\theta\to 0}\left[\sum_{(i,j)\in W^+}\frac{C_{ij}(\overline{x}+\theta e_{ij}) - C_{ij}(\overline{x})}{\theta} + \sum_{(i,j)\in W^-}\frac{C_{ij}(\overline{x}-\theta e_{ij}) - C_{ij}(\overline{x})}{-\theta}\right], \tag{2.5}$$

where W^+ is the set of arcs in W on which the flow increases and W^- the set of arcs on which the flow decreases. From (2.3) and (2.4), it follows that

$$C(W) = \sum_{(i,j)\in W^+} C^+_{ij}(\overline{x}) - \sum_{(i,j)\in W^-} C^-_{ij}(\overline{x}). \tag{2.6}$$

For an augmenting path P with respect to the flow $\overline{x}$, the augmenting cost $C(P)$ of the path P is the rate of change in the objective function as we augment the flow along the path P. Thus,

$$C(P) = \sum_{(i,j)\in P^+} C^+_{ij}(\overline{x}) - \sum_{(i,j)\in P^-} C^-_{ij}(\overline{x}), \tag{2.7}$$

where P^+ and P^- have the same meaning as W^+ and W^-.

2.3. The residual network. For the residual network $G(\overline{x}) = (N, A(\overline{x}))$ for a feasible flow $\overline{x}$ for every arc$(i,j) \in A$ with $\overline{x}_{ij} < u_{ij}$, we introduce an arc$(i,j) \in A(\overline{x})$ with residual capacity $r_{ij} = u_{ij} - \overline{x}_{ij}$ and cost $\hat{c}_{ij} = C^+_{ij}(\overline{x})$; for every arc$(i,j) \in A$ with $\overline{x}_{ij} > 0$, we introduce an arc$(j,i) \in A(\overline{x})$ with $r_{ji} = \overline{x}_{ij}$ and cost $\hat{c}_{ji} = C^-_{ij}(\overline{x})$.

Note that for the minimum cost flow problem, $\hat{c}_{ij} = c_{ij}$ if $(i,j) \in A$ and $\hat{c}_{ij} = -c_{ji}$ if $(j,i) \in A$. The network $G(\overline{x})$ contains cycles of the type $i-(i,j)-j-(j,i)-i$, that is, cycles consisting of arcs (i,j) and (j,i) only, if the flow $\overline{x}_{ij}$ lies in between the bounds, that is, $0 < \overline{x}_{ij} < u_{ij}$. We call these cycles as trivial cycles, since these cycles represent zero flow augmentation on the corresponding arcs. We note that any augmenting cycle W^1 in G with respect to the flow $\overline{x}$ corresponds to a directed cycle W in $G(\overline{x})$; the directed cycle W is a nontrivial cycle and is obtained by reversing the arcs in W^1 on which the flow is decreased. Conversely, any nontrivial directed cycle W in $G(\overline{x})$ corresponds to an augmenting cycle W^1 in G. The cost of a directed cycle W in $G(\overline{x})$ is defined by

$$C(W) = \sum_{(i,j)\in W} \hat{c}_{ij}. \tag{2.8}$$

Thus, by construction, the cost of the cycle W in $G(\overline{x})$ is the same as the cost of the corresponding augmenting cycle W^1 in G, that is, $C(W) = C(W^1)$. Henceforth, we mainly work with the residual network, since it simplifies the discussion and is equivalent to working with the original network.

2.4. Optimality conditions. We need the following two well-known conditions (for more details, see Rockafellar [6]) for a flow $\overline{x}$ to be optimal for SCCF. We state them using our notations and terminology.

Theorem 2.1 (negative cycle optimality conditions). *A feasible flow $\overline{x}$ to SCCF is optimal if and only if G contains no negative augmenting cycle W with respect to the flow $\overline{x}$ (equivalently, the residual network $G(\overline{x})$ contains no negative directed cycle).*

Theorem 2.2 (reduced cost optimality conditions). *A feasible flow $\overline{x}$ to SCCF is optimal if and only if there exists a node potential vector satisfying the following conditions:*

$$\hat{c}_{ij}^{\pi} \geq 0 \quad \forall (i,j) \in G(\overline{x}), \tag{2.9}$$

where

$$\hat{c}_{ij}^{\pi} = \hat{c}_{ij} - \pi(i) + \pi(j) \tag{2.10}$$

and $\hat{c}_{ij}$ are the costs of arcs in $G(\overline{x})$ as defined earlier.

The reduced cost optimality conditions in terms of the marginal costs $C_{ij}^{+}(\overline{x})$ and $C_{ij}^{-}(\overline{x})$ are

$$\begin{gathered}\pi(i) - \pi(j) \leq C_{ij}^{+}(\overline{x}) \quad \forall (i,j) \in A \text{ with } \overline{x}_{ij} = 0,\\ C_{ij}^{-}(\overline{x}) \leq \pi(i) - \pi(j) \leq C_{ij}^{-} + (\overline{x}) \quad \forall (i,j) \in A \text{ with } 0 < \overline{x}_{ij} < u_{ij},\\ C_{ij}^{-}(\overline{x}) \leq \pi(i) - \pi(j) \quad \forall (i,j) \in A \text{ with } \overline{x}_{ij} = u_{ij}.\end{gathered} \tag{2.11}$$

3. Arc tolerance analysis

In this paper, we investigate the problem of determining all possible convex cost perturbations of the cost function of an arc, while holding all other cost functions, supplies/demands, and capacities constant, such that a given solution to SCCF remains optimal. We refer to this problem as the arc tolerance analysis. In what follows, we develop a combinatorial theory for arc tolerance analysis based on the negative cycle optimality condition of Theorem 2.1. Let $\overline{x}$ be an optimal flow and let the cost function $C_{kl}(\overline{x})$, for arc(k,l), be perturbed. Let $\hat{C}_{kl}(\overline{x})$ denote the perturbed cost. We will say $\hat{C}_{kl}(\overline{x})$ is a valid perturbation of $C_{kl}(\overline{x})$ if it is convex and if $\overline{x}$ remains optimal when $C_{kl}(\overline{x})$ is replaced by $\hat{C}_{kl}(\overline{x})$, in the SCCF (2.1) and (2.2). From the construction of the residual network and the negative cycle optimality condition, it is clear that the factors which influence the optimality of the flow $\overline{x}$ with respect to the arc(k,l) are only the marginal costs of the cost function $C_{kl}(\overline{x})$. We are therefore able to characterize all valid perturbations $\hat{C}_{kl}(\overline{x})$ of $C_{kl}(\overline{x})$ in terms of the marginal costs of $\hat{C}_{kl}(\overline{x})$. We show that all valid perturbations of $C_{kl}(\overline{x})$ will either have

$$\hat{C}_{kl}^{+}(\overline{x}) \in [-\tilde{d}_l(k), \infty), \qquad \hat{C}_{kl}^{-} \in (-\infty, \tilde{d}_k(l)] \tag{3.1}$$

or

$$\hat{C}_{kl}^{+}(\overline{x}) \in [-\tilde{d}_l(k), \infty), \qquad \hat{C}_{kl}^{-} \in [-\infty, \tilde{d}_k(l)). \tag{3.2}$$

For the minimum cost flow problem with linear cost, we show that the arc tolerance set reduces to an interval on the real line. Let (k,l) denote an arbitrary but fixed arc, which is considered for finding its tolerance interval. Let $\widehat{C}_{kl}(x)$ be any convex function replacing $C_{kl}(x)$. The perturbed total cost function is

$$\widehat{C}(\overline{x}) = \sum_{(i,j)\in A,(i,j)\neq(k,l)} C_{ij}(\overline{x}) + \widehat{C}_{kl}(\overline{x}). \tag{3.3}$$

Let $\tilde{G}$ be the network obtained from $G(\overline{x})$ after deleting the arcs (k,l) and (l,k) if they are present in $G(\overline{x})$. From the construction of $G(\overline{x})$, we have

$$G(\overline{x}) = \begin{cases} \widetilde{G} \cup (k,l) & \text{if } \overline{x}_{kl} = 0, \\ \widetilde{G} \cup (l,k) & \text{if } \overline{x}_{kl} = u_{kl}, \\ \widetilde{G} \cup (k,l),(l,k) & \text{if } 0 < \overline{x}_{kl} < u_{kl}. \end{cases} \tag{3.4}$$

By the negative cycle optimality conditions, $G(\overline{x})$ contains no negative cycles. Furthermore, after perturbation, the costs of arcs (k,l) and (l,k) in $G(\overline{x})$ change to $\widehat{C}^{+}_{kl}(x)$ and $-\widehat{C}^{-}_{kl}(\overline{x})$, respectively, while costs of all other arcs remain unchanged. These two facts lead to the following observations.

Property 3.1. (a) $\widetilde{G}$ contains no negative cycles. (b) If, after perturbation, $G(\overline{x})$ contains a negative cycle, then the cycle contains either arc(k,l) or arc(l,k).

If a minimum cost cycle containing an arc is nonnegative, then any cycle containing the same arc is nonnegative. Thus Property 3.1(b) implies the following.

Property 3.2. After perturbation, $G(\overline{x})$ contains no negative cycles if and only if the following statements hold. (a) Any minimum cost cycle containing arc(k,l) in $G(\overline{x})$ has nonnegative cost. (b) Any minimum cost cycle containing arc(l,k) has nonnegative cost.

We denote the shortest distances from node p to node q in $\widetilde{G}$ by $\tilde{d}_p(q)$. We use the convention that $\tilde{d}_p(q) = \infty$ if there is no path from node p to node q.

Property 3.1(a) implies that one can find $\tilde{d}_p(q)$ in $\widetilde{G}$, in polynomial time using label correcting shortest path algorithms such as Bellman-Ford's. We will need the following property in subsequent discussions.

Property 3.3. $-\tilde{d}_l(k) \le \tilde{d}_k(l)$.

Proof. $\tilde{d}_k(l) + \tilde{d}_l(k)$ is the cost of a closed walk consisting of a shortest path from node k to node l and a shortest path from node l to node k in $\widetilde{G}$. Since $\widetilde{G}$ has no negative cycles, $\tilde{d}_k(l) + \tilde{d}_l(k) \ge 0$. In case no nonsingleton paths exist from l to k or k to l, the inequality is still valid since in that case the right-hand side is ∞. □

Lemma 3.4. *The convex function $\widehat{C}_{kl}(\overline{x})$ is a valid perturbation of $C_{kl}(\overline{x})$ if and only if the following conditions hold:*

(a) $\widehat{C}^{+}_{kl}(\overline{x}) \ge -\tilde{d}_l(k)$ *if* $\overline{x}_{kl} < u_{kl}$,
(b) $\hat{c}^{-}_{kl}(\overline{x}) \le \tilde{d}_k(l)$ *if* $\overline{x}_{kl} > 0$.

Proof. Any minimum cost cycle containing arc(k,l) is the union of arc(k,l) and a shortest path from node l to node k in $\widetilde{G}$, and its cost is $\tilde{C}_{kl}^{+}(\overline{x}) + \tilde{d}_l(k)$. Similarly, the cost of any minimum cost cycle containing arc(l,k) is $-\tilde{C}_{kl}^{+}(\overline{x}) + \tilde{d}_k(l)$. By negative cycle optimality conditions, the convex function $\tilde{C}_{kl}(x_{kl})$ is a valid perturbation if and only if, after the perturbation, $G(\overline{x})$ contains no negative cycles. By Property 3.2, this holds if and only if $\tilde{C}_{kl}^{+}(\overline{x}) + \tilde{d}_l(k) \geq 0$ if $(k,l) \in G(\overline{x})$, and $-\tilde{C}_{kl}^{-}(\overline{x}) + \tilde{d}_k(l) \geq 0$ if $(l,k) \in G(\overline{x})$. But $(k,l) \in G(\overline{x})$ if $\overline{x}_{kl} \leq u_{kl}$, and $(l,k) \in G(\overline{x})$ if $\overline{x}_{k,l} > 0$. Hence, the lemma. □

Note that a shortest path from node k to node l in $\widetilde{G}$ is a shortest path from node k to node l excluding the single arc path $k - (k,l) - l$, which is in turn a shortest augmenting path from node k to node l in G excluding the path $k - (k,l) - l$ or the path $k - (l,k) - l$. For this reason, $\tilde{d}_k(l)$ can be considered as the *nonsingleton shortest augmenting distance* from node k to node l in G.

We are now able to characterise the tolerance set for arc(k,l) in terms of the nonsingleton shortest augmenting distances, $\tilde{d}_l(k)$ and $\tilde{d}_k(l)$.

Theorem 3.5. *Let $\tilde{d}_k(l)$ and $\tilde{d}_l(k)$ denote the nonsingleton shortest augmenting distances with respect to the flow $\overline{x}$. Then $\hat{C}_{kl}(\overline{x})$ is a valid perturbation of $C_{kl}(\overline{x})$ if and only if*

(1) *for $\overline{x}_{kl} = 0$, $\hat{C}_{kl}^{+}$ or $\hat{C}_{kl}^{-} \in [-\tilde{d}_l(k), \infty)$,*
(2) *for $\overline{x}_{kl} = u_{kl}$, $\hat{C}_{kl}^{+}$ or $\hat{C}_{kl}^{-} \in (-\infty, \tilde{d}_l(k)]$,*
(3) *and for $0 < \overline{x}_{kl} < u_{kl}$, $\hat{C}_{kl}^{-} \in (-\infty, \tilde{d}_k(l)]$ and $\hat{c}_{kl}^{+} \in [-\tilde{d}_l(k), \infty)$.*

Proof

Sufficiency. consider the case when $\overline{x}_{kl} = 0$ and $\hat{C}_{kl}^{-} \in [-\tilde{d}_l(k), \infty)$. Since $\hat{C}_{kl}(x)$ is convex, $\hat{C}_{kl}^{+} \geq \hat{C}_{kl}^{-}$.

Hence, $\hat{C}_{kl}^{+} \in [-\tilde{d}_l(k), \infty)$, and condition (a) of Lemma 3.4 is satisfied. The case $\hat{C}_{kl}^{+} \in [-\tilde{d}_l(k), \infty)$ is trivial. Case (2) can be handled in the same way. For case (3) $\hat{C}_{kl}^{-} \in (-\infty, \tilde{d}_k(l)] \Rightarrow \hat{C}_{kl}^{-} \leq -\tilde{d}_l(k)$ and $\hat{C}_{kl}^{+} \in [-\tilde{d}_l(k), \infty) \Rightarrow \hat{C}_{kl}^{+} \geq -\tilde{d}_l(k)$.

Hence, both Lemma 3.4 (a) and (b) are satisfied.

Necessity. If both $\hat{C}_{kl}^{+}(\overline{x})$ and $\hat{C}_{kl}^{-}(\overline{x})$ lie outside $[-\tilde{d}_l(k))$ when $\overline{x}_{kl} = 0$, it implies that $\hat{C}_{kl}^{+}(\overline{x}) \leq -\tilde{d}_l(k)$ and condition (a) of Lemma 3.4 is violated. Hence, $\hat{C}_{kl}(\overline{x})$ is not a valid perturbation. The other two cases can be treated in the same way. □

For the minimum cost flow problem with linear cost functions, that is, $C_{ij}(\overline{x}) = C_{ij}\overline{x}_{ij}$ for all $(i,j) \in A, C_{kl}^{+}(\overline{x}) = C_{kl}^{-}(\overline{x}) = C_{kl}$, and the tolerance set in this case will reduce to one of the following three intervals: $(-\infty, \tilde{d}_k(l)]$, $[-\tilde{d}_l(k), \infty)$, or $[-\tilde{d}_l(k), \tilde{d}_k(l)]$.

By the construction of these intervals, it follows that any one of them will constitute an exact tolerance interval for the minimum cost flow problem.

4. Algorithm for finding nonsingleton shortest paths

In Section 3, we have shown that the tolerance set of an arc(k,l) can be immediately calculated, once we know nonsingleton shortest distances $\tilde{d}_k(l)$ and $\tilde{d}_l(k)$ in $G(\overline{x})$ (Theorem 3.5). Since $G(\overline{x})$ contains no negative cycles, but contains usually many negative arc costs, we can find tolerance sets for all arcs by applying a label correcting shortest path algorithm (such as Bellman and Ford's algorithm) in $O(m)$ times. The best (strongly) polynomial time bound for any label correcting algorithm is $O(nm)$ (Ahuja et al. [1, Chapter V].) This straightforward method for computing all relevant nonsingleton shortest distances would be highly inefficient, since its complexity would be $O(nm^2)$. In this section, we improve this complexity significantly to $O(n^3)$ in two ways:

(1) by working with optimal node potentials,
(2) by reoptimization.

4.1. Working with an optimal node potential. The reduced cost optimality conditions (Theorem 2.2) guarantee that there exist node potentials π such that $\hat{c}_{ij}^{\pi} \geq 0$ for all arcs in $G(\overline{x})$. We call such a π an *optimal node potential* vector. From definition of $\hat{c}_{ij}^{\pi}$, it follows that for any directed path P from node p to node q,

$$\sum_{(i,j)\in P} \hat{c}_{ij}^{\pi} = \sum_{(i,j)\in P} \hat{c}_{ij} - (\pi(p) - \pi(q)). \tag{4.1}$$

From (4.1), it follows that if we denote the nonsingleton shortest distance with respect to $\hat{c}_{ij}^{\pi}$'s from node p to node q by $\tilde{d}_p^{\pi}(q)$, then

$$\tilde{d}_p(q) = \tilde{d}_p^{\pi}(q) + \pi(p) - \pi(q). \tag{4.2}$$

We often get a node potential vector π satisfying $\hat{c}_{ij}^{\pi} \geq 0$ for every (i,j) in $G(\overline{x})$, as a by-product of solving SCCF. Otherwise, we use the following procedure to find such a node potential vector. The procedure has the following three steps.

(1) Augment the network $G(\overline{x})$ by adding a node s to $G(\overline{x})$ and connect this node to every node i by an arc(s,i) of cost 0.

(2) Find shortest distance $d(i)$ from node s to every node i in the augmented network by applying a label correcting shortest path algorithm.

(3) Let $\pi = -d$.

Step (2) dominates all other steps and, as we pointed out at the beginning of this section, requires $O(nm)$ time. We call it as procedure *potential-initialization*.

4.2. The reoptimization procedure. Once we have optimal potentials at hand, we proceed as follows. We choose a node p and apply Dijkstra's algorithm to find a shortest path tree T_p from node $p \in G(\overline{x})$, using arc costs $\hat{c}_{ij}^{\pi}$. The path from node p to any node i in T_p is a shortest path from node p to node i, and we denote the corresponding shortest distance by $d_p^{\pi}(i)$. For each node i, the predecessor index pred(i) gives the predecessor node j of node i in T_p. We will say that pred$(p) = 0$. Using index pred(i) we construct other tree indices, namely, depth index and thread index of each node $i \in N$, denoted by depth(i) and thread(i). For details, refer to Ahuja et al. [1, Chapter 11]. The index depth(i) stores the number of arcs in the path from the root node p to node i in the tree. The indices thread(i)'s define a traversal of the tree, starting from the root node p, and visiting other nodes in a "top-to-bottom" and "left-to-right" manner, and finally returning to the root node. For node i, thread(i) is the node in depth-first search encountered just after node i.

First we build adjacency list of each node i in T_p, denoted by SUCC(i). We then apply breadth-first and depth-first search in T_p and use SUCC(i)'s, to construct depth indices and thread indices, respectively, the procedure takes $O(n)$ time. We will see below how these indices are used for efficient reoptimization.

We first point out that the tree paths in T_p from p to all nodes except for those in SUCC(p) are nonsingleton shortest paths. To find a nonsingleton shortest path from p to node q in SUCC(p), we delete the arc(p,q) from T_p as well as from $G(\overline{x})$; as a result, all relevant information, pred(i)'s and $d_p^{\pi}(i)$'s, becomes invalid only for nodes in $D(q)$, the set of all descendants of node q including q. Let, for each node $i \in D(q)$, nsd(i) denote the distance of a shortest path from node p to node i in $G(\overline{x}) - (p,q)$ passing through the set of permanently labeled nodes (i.e., the nodes for which shortest distances have already been found). Note that $N - D(q)$ is the set of permanently labeled nodes just after the deletion of arc(p,q). Let nspread(i) denote the predecessor of node i in the corresponding shortest path. We initialize the label nsd(j) for each node $j \in D(q)$ as follows:

$$\operatorname{nsd}(j) = \min d_p^{\pi}(i) + \hat{c}_{ij}^{\pi} : (i,j) \in B(j) : i \in N - D(q), \tag{4.3}$$

where $B(j)$ is the backward star of node j. To perform this process efficiently, we first mark nodes in $D(q)$; we do this by tracing thread(i) starting from node q until the depth of node next traced is at least equal to that of node q (we use depth indices). Then, we again visit nodes $D(q)$ one by one using thread indices, and scan their backward star to initialize nsd(j) for each $j \in D(q)$. The algorithmic description of this procedure is given under procedure *setup-potentials* ($D(q)$). We now apply Dijkstra's algorithm taking $S = N - D(q)$ as the initial set of permanently labeled nodes; for $i \in D(q)$, we use the label nsd(i) instead of $d_p^{\pi}(i)$. At the termination of the iterative loop, nsd(q) is the nonsingleton shortest distance from p with respect to costs $\hat{c}_{ij}^{\pi}$. We call this part of the reoptimization as *update* ($D(q)$), whose description is given under procedure *update* ($D(q)$).

The procedure setup-potentials initializes the distance labels nsd(i) to the shortest distance to node i passing only through nodes in $s = N - D(q)$ in the graph $G(\overline{x}) - (p,q)$, for each node $i \in D(q)$. Thus we can apply Dijkstra's algorithm taking $S = N - D(q)$ as the initial set of permanently labeled nodes. Hence, at the termination of reoptimization process, nsd(q) is the shortest distance in $G(\overline{x}) - (p,q)$, that is, the nonsingleton shortest distance from node p to node q.

The validity of the procedure update $D(q)$ follows from the correctness of Dijkstra's algorithm. We now analyze the complexity of the reoptimization process. We refer to the reoptimization process done after deleting the arc(p,q) as reoptimization (p,q). It has two steps:

(i) setting initial potentials for nodes in $D(q)$,
(ii) applying Dijkstra's algorithm on the graph $G(x) - (p,q)$ taking $S = N - D(q)$ as the initial set of permanently labelled nodes.

For step (i), marking nodes in $D(q)$ takes $|D(q)|$ time; scanning the backward star of each node in $D(q)$ takes $O(\sum_{i\in D(q)} |B(i)|)$ time. In step (ii), scanning forward stars of nodes in $D(q)$ takes $O(\sum_{i\in D(q)} |F(i)|)$ time. Selecting the node with the minimum

```
Procedure setup-potentials ($D(q)$);
begin
        $\text{mark}(q) = q$; $\text{nsd}(q) = \infty$;
        $k := \text{thread}(q)$;
while $\text{depth}(k) > \text{depth}(q)$ do
        $\text{mark}(k) = q$, $\text{nsd}(k) := \infty$ and $k := \text{thread}(k)$;
        for each $(i,q) \in B(q)$ do
            if $\text{mark}(i) \neq q$ and $\text{nsd}(q) > d_p^\pi(i) + \hat{c}_{ij}^\pi$, then
              $\text{nsd}(q) := \text{nsd}(q) > d_p^\pi(i) + \hat{c}_{ij}^\pi$ and $\text{nspread}(q) := i$;
              $k = \text{thread}(q)$;
        while $\text{depth}(k) > \text{depth}(q)$ do
        begin
            for each $(i,k) \in B(k)$ do
            begin
                if $\text{mark}(i) \neq q$, then
                  if $\text{nsd}(k) > d_p^\pi(i) + \hat{c}_{ij}^\pi$, then
                    $\text{nsd}(k) > d_p^\pi(i) + \hat{c}_{ij}^\pi$ and $\text{nspred}(k) := i$;
            end;
            $k := \text{thread}(k)$;
        end;
end.
```

Algorithm 4.1

distance label takes $O(|D(q)|)$ time, if we store nsd(i)'s of nodes in $D(q)$ in an array (we can use thread indices to scan the nodes in $D(q)$ only); hence, this takes $O(|D(q)|^2)$ in total. We summarize this discussion in the following lemma.

Lemma 4.1. *The procedure reoptimization* (p,q) *runs in* $O\sum_{i\in D(q)}(|F(i)|+|B(i)|+|D(q)|^2)$.

4.3. The algorithm. Having developed the required subroutines already, we now describe the algorithm for computing nonsingleton shortest path between all pairs of nodes. We assume that right and left marginal costs of each arc are given or can be computed in $O(1)$ time. The algorithm has the following steps.

(i) Find an optimal node potential vector π, if it is not available.

(ii) For each node $p \in N$ by Algorithms 4.1 and 4.2, find nonsingleton shortest distances from p to all other nodes using arc costs c_{ij}^π.

(iii) Using the formula $\tilde{d}_i(j) = d_i^\pi(j) + \pi(i) - \pi(j)$, calculate nonsingleton shortest distances between all pairs of nodes.

Step (ii) consists of the following steps for each node p:

(iia) find a shortest path tree T_p from node p to all other nodes (using Dijkstra's algorithm);

(iib) build tree indices, thread and depth, for the tree T_p;

(iic) for each $q \in \text{SUCC}(P)$, use procedure reoptimization (p,q) to find nonsingleton shortest distance $d_p^\pi(q)$ from p to node q.

```
Procedure $update$ $(D(q))$;
begin
    $S = N - D(q), \overline{S} = D(q)$;
    while $|S| < |N|$ do
    begin
        let $i \in \overline{S}$ be a node for which $\text{nsd}(i) = \min\{\text{nsd}(j) : j \in \overline{S}\}$;
        $S = S \cup i$;
        $\overline{S} = S - i$;
        for each $(i, j) \in A(i)$ and $j \in \overline{S}$ do
        if $\text{nsd}(j) > \text{nsd}(i) + \hat{c}_{ij}^{\pi}$, then $\text{nsd}(j) := \text{nsd}(i) + c_{ij}$
        and $\text{nspred}(j) = i$;
    end;
end.
```

Algorithm 4.2

Recall that, for nodes not in SUCC(p), the unique tree path in T_p is not only the shortest but also the nonsingleton shortest path. The steps of the algorithm are self-explanatory, except for reoptimization (p,q) (in step (iic) for which we have already given the justification). Step (i) (finding an optimal potential) requires $O(nm)$ time. Step (iii) requires $O(n^2)$ time. We now analyze the complexity of step (ii) for node, say, p; multiplying this complexity by n gives the complexity of step (ii). Using Dijkstra's algorithm, finding a shortest path tree T_p, step (iia) requires $O(n^2)$ time (see, e.g., Ahuja et al. [1]). Building tree indices for T_p, step (iib) requires $O(n)$ time. In step (iic), we apply procedure reoptimization (p,q) for each node $q \in$ SUCC(p). By Lemma 4.1, procedure reoptimization (p,q) requires $O(\sum_{i \in D(q)}(|B(i)| + |F(i)| + |D(q)|^2))$ time. Note that $U_{q \in \text{SUCC}(p)} D(q) = N - p$. Thus summing the expression $\sum_{i \in D(q)}(|B(i)| + |F(i)|)$ over all q in SUCC(p), we get the bound $\sum_{i \in N}(|B(i)| + |F(i)|) = m + m = O(m)$. Similarly, the summation of $|D(q)|^2$ over all q in SUCC(p) gives $O(n^2)$ bound. (Here we use the property that if $a_1, a_2, \ldots, a_k \geq 0$, $(a_1 + a_2 + \cdots + a_k)^2 > \sum_{i=l}^{k} a_i^2$ and the fact that $\sum_{q \in \text{SUCC}(p)} |D(q)| < n$). Thus, summing complexity bounds for procedure reoptimization (p,q) given in Lemma 4.1, we get the complexity of step (iic) for a node p as $O(m + n^2)$. Since the complexity of this step dominates the complexity of all the other steps, the time complexity of step (ii) for node p is $O(n^2)$. Hence, step (ii) (over all nodes) requires $O(n^3)$ time. Since this step dominates the bound for step (i) and step (iii), the algorithm runs in $O(n^3)$ time. We summarize the discussion in the following theorem.

Theorem 4.2. *The algorithm given in this section finds nonsingleton shortest distances $G(\overline{x})$ between all pairs of nodes in $O(n^3)$.*

Apart from the storage requirement for the initial data, for every node p, we need an additional n-dimensional array for storing each of the following data: distance labels $d_p^{\pi}(i)$, three indices pred(i), depth(i), thread(i). Thus, over all nodes, we require $O(n^2)$ additional storage.

Theorem 4.2 together with Theorem 3.5 implies that one can find tolerance sets of all arcs for a separable convex cost flow (SCCF) problem in $O(n^3)$ time.

5. Concluding remarks

We have developed a theory for cost perturbations of arc cost functions for the separable convex cost flow problem. For each arc, these perturbations are associated with a set, which we call as the *tolerance set*; this set is obtained from nonsingleton shortest distances between the head and tail nodes of the corresponding arc. For the minimum cost flow problem, this tolerance set reduces to an exact tolerance interval without having to contract the original network. The algorithm we have developed computes tolerance sets for all arcs in $O(n^3)$ time in array implementation, which essentially computes nonsingleton shortest distances between all pairs of nodes. Using Fibonacci heaps developed by Fredman and Tarjan [3] instead of arrays, our algorithm will run in $O(nm+n^2\log n)$ time. Details can be found in [8]. The assumption that the left and right derivatives of arc cost functions can be computed in $O(1)$ time is valid for piecewise linear convex functions, quadratic convex functions, and linear functions.

Acknowledgment

We wish to express our gratitude to Professor R. K. Ahuja for valuable suggestions.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows. Theory, Algorithms, and Applications*, Prentice Hall, New Jersey, 1993.
[2] C. O. Fong and V. Srinivasan, *Determining all nondegenerate shadow prices for the transportation problem*, Transportation Sci. **11** (1977), 199–222.
[3] M. L. Fredman and R. E. Tarjan, *Fibonacci heaps and their uses in improved network optimization algorithms*, J. Assoc. Comput. Mach. **34** (1987), no. 3, 596–615.
[4] G. Maier, *Determining all nondegenerate bounds in sensitivity analysis of minimal-cost network-flow-problems*, 11th Symposium on Operations Research (Darmstadt, 1986), Methods of Operations Research, vol. 57, Athenäum/Hain/Hanstein, Königstein, 1987, pp. 145–153.
[5] W. B. Powell, *A review of sensitivity results for linear networks and a new approximation to reduce the effects of degeneracy*, Transportation Sci. **23** (1989), no. 4, 231–243.
[6] R. T. Rockafellar, *Network Flows and Monotropic Optimization*, Pure and Applied Mathematics, John Wiley & Sons, New York, 1984.
[7] D. R. Shier and C. Witzgall, *Arc tolerances in shortest path and network flow problems*, Networks **10** (1980), no. 4, 277–291.
[8] P. T. Sokkalingam, *The minimum cost flow problem: primal algorithms and cost perturbations*, Ph.D. thesis, Indian Institute of Technology, Kanpur, 1995, unpublished.
[9] V. Srinivasan and G. L. Thompson, *An operator theory of parametric programming for the transportation problem. I, II*, Naval Res. Logist. Quart. **19** (1972), 205–225, 227–252.

P. T. Sokkalingam: HCL Technologies Ltd. CODC, Chennai 600 026, India
E-mail address: sokk@rediffmail.com

Prabha Sharma: Department of Mathematics and Statistics, Indian Institute of Technology Kanpur, Kanpur 208 016, India
E-mail address: prabha@iitk.ac.in