JOURNAL OF INDUSTRIAL AND
MANAGEMENT OPTIMIZATION
Volume **5**, Number **4**, November **2009**

doi:10.3934/jimo.2009.5.929

pp. **929–950**

# A NETWORK SIMPLEX ALGORITHM FOR SOLVING THE MINIMUM DISTRIBUTION COST PROBLEM

I-Lin Wang and Shiou-Jie Lin

Department of Industrial and Information Management
National Cheng Kung University
Tainan, 701, Taiwan

(Communicated by Shu-Cherng Fang)

Abstract. To model the distillation or decomposition of products in some manufacturing processes, a minimum distribution cost problem (MDCP) for a specialized manufacturing network flow model has been investigated. In an MDCP, a specialized node called a D-node is used to model a distillation process that connects with a single incoming arc and several outgoing arcs. The flow entering a D-node has to be distributed according to a pre-specified ratio associated with each of its outgoing arcs. This proportional relationship between arc flows associated with each D-node complicates the problem and makes the MDCP more difficult to solve than a conventional minimum cost network flow problem. A network simplex algorithm for an uncapacitated MDCP has been outlined in the literature. However, its detailed graphical procedures including the operations to obtain an initial basic feasible solution, to calculate or update the dual variables, and to pivot flows have never been reported. In this paper, we resolve these issues and propose a modified network simplex algorithm including detailed graphical operations in each elementary procedure. Our method not only deals with a capacitated MDCP, but also offers more theoretical insights into the mathematical properties of an MDCP.

1. **Introduction.** The minimum cost flow problem is a specialized linear programming problem with network structure which seeks an optimal flow assignment over a network satisfying the constraints of node flow balance and arc flow bounds (see [1]). However, these constraints are too simplified to model some real cases such as, for example, the synthesis and distillation of products in some manufacturing processes. For this purpose, Fang and Qi [8] proposed a generalized network model called the *manufacturing network flow* (MNF). The MNF considers three specialized nodes: I-nodes, C-nodes, and D-nodes, to model the nodes of inventory, synthesis (combination), and distillation (decomposition), in addition to the conventional nodes: S-nodes, T-nodes, and O-nodes, which serve as sources, sinks, and transhipment nodes, respectively. Fang and Qi [8] also defined a *minimum distribution cost problem* (MDCP) for a specialized MNF model referred to as the distribution network which contains both D-nodes and conventional nodes. A D-node represents a distillation process and only connects with a single incoming arc

2000 *Mathematics Subject Classification.* Primary: 90B10, 05C85; Secondary: 05C21.

*Key words and phrases.* network optimization, manufacturing network, distribution network, minimum distribution cost flow problem, network simplex algorithm.

I-Lin Wang was partially supported by the National Science Council of Taiwan under Grant NSC95-2221-E-006-268.

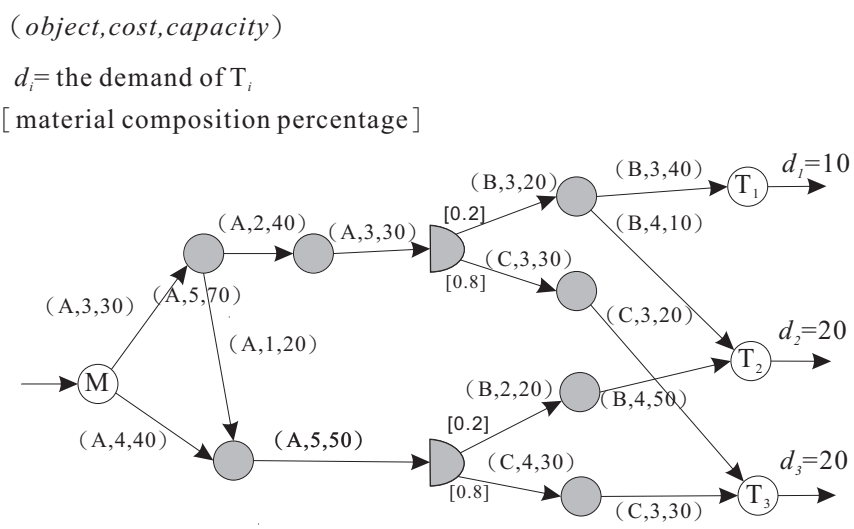



FIGURE 1. An MDCP example in a supply chain network

and several outgoing arcs. The flows passing through a D-node have to satisfy the *flow distillation constraint*. This constraint requires the flows entering a D-node to be distributed to each of its outgoing arcs called *distillation arcs*, according to a pre-specified ratio associated with each outgoing arc. For each D-node $i$ with entering flow $x_{i^*i}$ on its incoming arc $(i^*, i)$, the flow distillation constraint specifies the flow on its distillation arc $(i, j)$ to be $x_{ij} = k_{ij}x_{i^*i}$ where $k_{ij}$ is a constant between 0 and 1 and $\sum_{(i,j)} k_{ij} = 1$.

The MDCP can appear often in part of a supply chain network. Take Figure 1 as a reverse logistics network example. A recycled product collecting site $M$ distributes a recycled product $A$ which then will be decomposed (or distilled) into materials $B$ and $C$ in two different recycle plants (half-circled nodes), and finally transported to landfills $T_1$, $T_2$ , and $T_3$. In this example, each recycle plant is a D-node; $M$ is the S-node; $T_1$, $T_2$, and $T_3$ are the T-nodes; and all the other nodes are O-nodes. Suppose that each recycle plant can decompose one unit of $A$ into 0.2 unit of $B$ and 0.8 unit of $C$. Each arc in the reverse logistics network in Figure 1 represents some operation (e.g. transportation) associated with a unit cost as well as a capacity constraint. The MDCP in Figure 1 seeks the minimum possible total cost to satisfy the demand requirements of landfills $T_1$, $T_2$, and $T_3$.

At first glance, the MDCP seems similar to the generalized network flow problem where the flow along an arc $(i, j)$ may lose or gain based on a gain factor $\mu_{ij}$, defined as the ratio of the flow arriving at the head node $j$ to the flow leaving from the tail node $i$. In fact, the generalized flow problem is a special case of the MDCP model. Take Figure 2(a) as an example. For each arc $(i, j)$ with loss of flows (i.e. $\mu_{ij} < 1$), one may convert it to an MDCP model by adding a dummy sink arc $(i, j')$ and a D-node $\alpha$ where the flow distillation factors $k_{\alpha j}$ and $k_{\alpha j'}$ can be derived from $\mu_{ij}$. Similarly, one may also convert each arc with the gain of flows to an MNF model by adding a dummy source arc and a C-node (see Figure 2(b) for details). Besides the generalized flow problem, Cohen and Megiddo [6] also discussed a class of parametric flow problems in which the fixed ratio flow problem is similar to the

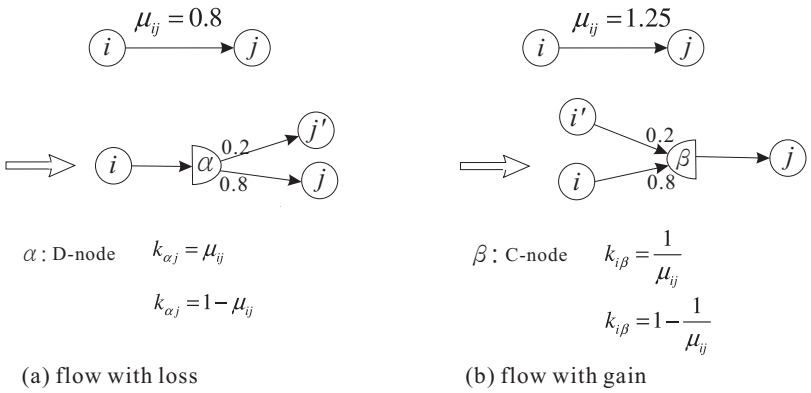



FIGURE 2. Converting a generalized flow to the MNF model

MDCP. In particular, the fixed-ratio flow problems have special constraints which are similar to the distillation constraints of the MDCP. Nevertheless, the result in [6] does not help much because of the following two reasons: First, the fixed ratio flow problem can be solved in polynomial time only when the number of those special constraints is a constant, which is more restrictive than MDCP; Second, Cohen and Megiddo [6] only showed the existence of a strongly polynomial algorithm through a sequence of reduction from other problems, rather than providing its detailed operations. Therefore, MDCP is more general and difficult than both the generalized flow problem and the fixed ratio flow problem in the literature.

Although MDCP was defined by Fang and Qi [8], similar problems have been studied by Koene [16], Chen and Engquist [5], and Chang et al. [4] in 1980s. In particular, the *pure processing network* introduced by Koene [16] also contains nodes of refining, transportation, and blending processes, and is the same as the manufacturing network here. Koene [16] showed that any given LP problem can be transformed to a pure processing network problem, and thus can be solved by his specialized primal algorithms. Chen and Engquist [5] proposed basis partitioning techniques in their primal simplex algorithm that used specialized network data structure for the network portion of the problem while treating the other non-network portion as either side variables or constraints. Chang et al. [4] further improved the practical efficiency for the algorithms by Chen and Engquist [5]. Although these previous works also focused on solving MDCPs, their solution methods still involve more algebraic operations, which are very different from the graphical operations proposed in this paper. Recently, Lu et al. [17] studied the generalized MDCP, where $\sum_{(i,j)} k_{ij}$ for all arcs emanating from a D-node $i$ may not necessarily equal to 1. They proposed modified network simplex algorithms to deal with min-cost flow problems on a generalized processing network, which has also been previously investigated by Koene [16].

The flow distillation constraints associated with each D-node complicate the problem and make the MDCP harder than the conventional minimum cost network flow problem. Although Fang and Qi [8] proposed a network simplex algorithm for solving the MDCP, their method is, however, not complete in the sense that it only describes the algebraic structure for a basis. Their proposal lacks details for the graphical operations such as obtaining the initial solution, and the pivoting

and updating procedures for both the arc flows and node potentials. This paper resolves these issues and proposes a network simplex algorithm with detailed graphical operations for solving an MDCP.

In addition to the MDCP, Fang and Qi [8] also introduced a max-flow problem for a distribution network, but they did not propose any method to solve the problem. Sheu et al. [19] proposed a multi-labelling method to solve this problem by adopting the concept of the augmenting path method [9] and the Depth-First Search algorithm (DFS) which tries every augmenting subgraph that goes to the sink or source and satisfies the flow distillation constraints. After finding such an augmenting subgraph, the algorithm then identifies decomposable components in an augmenting subgraph where the flow inside each component can be represented by a single variable. The flow can be calculated by the flow balance constraints for all nodes that joint different components. Although this method is straightforward, its complexity is shown to be non-polynomial.

Wang and Lin [21] proposed compacting rules and a polynomial-time compacting algorithm which can serve as a preprocessing procedure to simplify the MDCP problem structure. Specifically, a group of connected D-nodes can be shrunk into a single D-node. Any transhipment O-node with single incoming and outgoing arcs can be transformed into a single arc. Capacities on arcs connecting with a D-node can be unified using the same standards by its flow distillation constraints. After conducting their polynomial-time compacting procedures, the original network will be compacted to an equivalent one of smaller size.

Wang and Yang [22] solved three specialized uncapacitated MDCPs: $\text{UMDCP}_1$, $\text{UMDCP}_2$, and $\text{UMDCP}_3$ based on Dijkstra's algorithm [7]. Other network compacting rules which unify the effects of arc costs have also been proposed by the same author. Although two algorithms modified from the Dijkstra's algorithm have been designed to solve $\text{UMDCP}_1$ and $\text{UMDCP}_2$, they can not efficiently solve $\text{UMDCP}_3$.

A multicommodity MNF model of $\sigma + 1$ commodities was investigated by Mo et al. [18], in which there are $\sigma + 1$ layers with each layer corresponding to a network of the same commodity with inter-layer arcs connecting C-nodes or D-nodes. Their model is more restrictive and simplified in the sense that: First, it is assumed that there is only one kind of D-node to decompose commodity 0 into $\eta$ commodities and one kind of C-node to combine commodity $\sigma$ from $\lambda$ commodities; Second, the distillation factor $k$ associated with each D-node (or C-node) is assumed to be identical (i.e. $k = 1/\eta$ for a D-node, or $k = 1/\lambda$ for a C-node). Moreover, the elementary procedures in their proposed network simplex algorithm are still pure algebraic rather than graphical operations.

Among all the related literatures, the network-simplex-based solution method by Venkateshan et al. [20] discussed more graphical data structures and operations for solving min-cost MNF problems. Their works are similar to this paper, but their algorithm and notations were not clearly presented. On the other hand, we give more detailed and clear graphical illustration on the theoretical characteristics and complexity for each step of our network simplex algorithm.

In short, several specialized MDCPs have been solved in the literature, yet their problems are more restrictive (e.g. [18], [21], [22], [19]) or lack graphical operations (e.g. [16], [5], [4], [8], [18]). Since a network simplex algorithm should contain more graphical operations than pure algebraic operations like the conventional simplex algorithm, we propose here in this paper the detailed graphical operations for each elementary procedure in a network simplex algorithm for solving the MDCP. Our

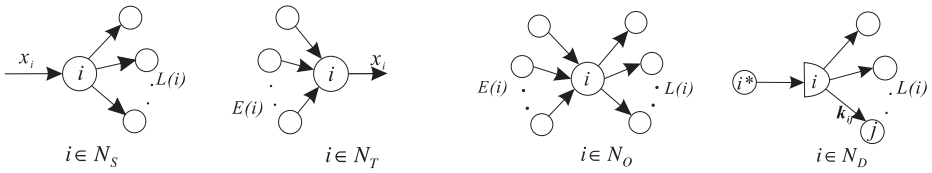



FIGURE 3. S-node, T-node, O-node and D-node

work provides a more efficient graphical implementation and more insights into the solution structures. The techniques developed in this paper can also be used to solve the problems in [19] and [22].

The rest of this paper is organized as follows: Section 2 introduces definitions and notations, presents a model transformation for easier illustration, defines a basic feasible graph, and provides optimality conditions for our network simplex algorithm. Detailed procedures of our network simplex algorithm are illustrated in Section 3. Finally, Section 4 summarizes and concludes the paper.

## 2. **Preliminaries.**

### 2.1. **Notations and model transformation.**

Let $G = (N, A)$ be a directed simple graph with node set $N$ and arc set $A$. For each arc $(i, j) \in A$, we associate it with a unit flow cost $c_{ij}$ and a flow capacity $u_{ij}$ where the arc flow $x_{ij} \in [0, u_{ij}]$. For each node $i \in N$, we define the set of nodes connecting to and from it as $E(i) := \{j \in N : (j, i) \in A\}$ and $L(i) := \{j \in N : (i, j) \in A\}$, respectively. There are four kinds of nodes (see Figure 3): S-nodes, T-nodes, O-nodes, and D-nodes, and they are denoted by $N_S$, $N_T$, $N_O$, and $N_D$, respectively. An S-node is a source node connected only by outgoing arcs. A T-node denotes a sink node connected only by incoming arcs. An O-node represents a transshipment node connected with both incoming and outgoing arcs. Usually, an S-node is a supply node, a T-node is a demand node, and an O-node is for transshipment. We refer to these three types of nodes as conventional nodes since they only have to satisfy the flow balance constraints. A D-node connects with one incoming arc and at least two outgoing arcs. For each node $i \in N_D$ with incoming arc $(i^*, i)$, the flow distillation constraint specifies $x_{ij} = k_{ij} x_{i^* i}$ for the flow on its distillation arc $(i, j)$. Furthermore, we assume $\sum_{j \in L(i)} k_{ij} = 1$ in order to satisfy the flow balance constraint. Without loss of generality, we assume all the networks in this paper to have already been compacted using the rules and algorithms by Wang and Lin [21] and Wang and Yang [22]. Also, we assume that the MDCP problem contains a finite optimal solution.

Shipping flows from several S-nodes to several T-nodes through O-nodes and D-nodes, an MDCP model proposed by Fang and Qi [8] is defined as follows:

$$\min \sum_{i \in N_S} c_i x_i + \sum_{(i,j) \in A} c_{ij} x_{ij} \qquad (\text{MDCP}_{FQ})$$

$$s.t. \quad \sum_{j \in L(i)} x_{ij} - x_i = 0 \quad \forall i \in N_S \qquad (1)$$

$$x_j - \sum_{i \in E(j)} x_{ij} = 0 \quad \forall j \in N_T \tag{2}$$

$$\sum_{j \in L(i)} x_{ij} - x_{i^* i} = 0 \quad \forall i \in N_D, (i^*, i) \in A \tag{3}$$

$$x_{ij} - k_{ij} x_{i^* i} = 0 \quad \forall i \in N_D, (i^*, i) \in A, (i, j) \in A \tag{4}$$

$$x_i \leq u_i \quad \forall i \in N_S \tag{5}$$

$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A \tag{6}$$

where $u_i$ for each $i \in N_S$ represents the maximum possible flow that a source node $i$ receives for shipping out, $d_j$ for each node $j \in N_T$ denotes the minimum amount of flow that a sink node has to receive, and $\sum_{i \in N_S} u_i \geq \sum_{j \in N_T} d_j$.

Although this formulation is straightforward, it does not follow the conventional notations as they usually appear in the literature of network optimization. For example, $x_i$ for each $i \in N_S$ and $x_j$ for each $j \in N_T$ can not be treated as arc flows since the incoming arc for each S-node and the outgoing arc for each T-node have not been defined in the model. Moreover, each node in a conventional minimum cost network flow model usually has an exact amount of supply or demand, rather than a lower bound (e.g. T-nodes) or upper bound (e.g. S-nodes) as defined in MDCP$_{FQ}$. To avoid confusion caused by different notations, we provide the following modifications for MDCP$_{FQ}$, based on the modeling techniques proposed in [13, 11, 10, 12, 3, 2, 14, 1, 15]:

1. Add a dummy source node $s$, an arc $(s, i)$ with $c_{si} = 0$, $x_{si} = x_i$, and $u_{si} = u_i$ for each S-node $i$. Assign $\sum_{i \in N_S} u_i$ and 0 units of supply for $s$ and each node $i \in N_S$, respectively.
2. Add a dummy sink node $t$, an arc $(j, t)$ with $c_{jt} = 0$, $x_{jt} = x_j - d_j$, and $u_{jt} = \infty$ for each T-node $j$. Assign $\sum_{i \in N_S} u_i - \sum_{j \in N_T} d_j$ and $d_j$ units of demands for $t$ and each node $j \in N_T$, respectively.
3. Add the arc $(s, t)$ with $c_{st} = 0$ and $u_{st} = \infty$.

Our modification successfully transforms MDCP$_{FQ}$ into a formulation similar to the conventional minimum cost network flow model where each node has an exact amount of supply or demand and each flow variable is associated with an arc containing both a head node and a tail node. In particular, as illustrated in Figure 4, each S-node and T-node, as well as the dummy nodes $s$ and $t$, all have a fixed amount of net flow. The modified formulation thus contains only two types of nodes: (1) the D-nodes, and (2) the other nodes denoted as the $\widehat{O}$-nodes, including the original S-nodes, T-nodes, O-nodes, and the dummy nodes $s$ and $t$. Let $N_{\widehat{O}} = N_S \cup N_O \cup N_T \cup \{s, t\}$, then we update $N = N_{\widehat{O}} \cup N_D$ and $A = A \cup \{(s, t)\} \cup \{(s, i) : i \in N_S\} \cup \{(j, t) : j \in N_T\}$. The new MDCP formulation after our transformation can be described as follows:

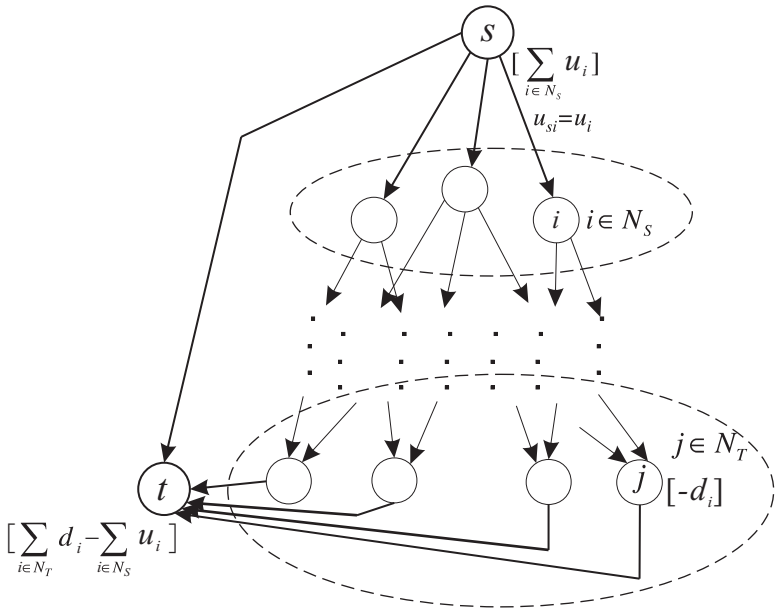



FIGURE 4. Transforming the MDCP$_{FQ}$ to the conventional minimum cost network flow model

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{MDCP}$$

$$\text{s.t.} \quad x_{st} + \sum_{i\in N_S} x_{si} = \sum_{i\in N_S} u_i \quad \text{for } s \tag{7}$$

$$x_{si} - \sum_{j\in L(i)} x_{ij} = 0 \quad \forall i \in N_S \tag{8}$$

$$\sum_{j\in L(i)} x_{ij} - \sum_{j\in E(i)} x_{ij} = 0 \quad \forall i \in N_O \tag{9}$$

$$x_{jt} - \sum_{i\in E(j)} x_{ij} = -d_j \quad \forall j \in N_T \tag{10}$$

$$-x_{st} - \sum_{j\in N_T} x_{jt} = \sum_{j\in N_T} d_j - \sum_{i\in N_S} u_i \quad \text{for } t \tag{11}$$

$$\sum_{j\in L(i)} x_{ij} - \sum_{j\in E(i)} x_{ij} = 0 \quad \forall j \in N_D \tag{12}$$

$$k_{ij}x_{i^*i} - x_{ij} = 0 \quad \forall i \in N_D, (i^*, i) \in A, (i, j) \in A \tag{13}$$

$$0 \le x_{ij} \le u_{ij} \quad \forall (i, j) \in A \tag{14}$$

where equations (7) to (12) define the flow balance for each node in $N$, equation (13) is the flow distillation constraint associated with each D-node, and equation (14) defines the arc flow bounds. Note that we assume $\sum_{j\in L(i)} k_{ij} = 1$ which means that equation (12) can be derived from equation (13) and thus equation (12) is removable. This new formulation has several advantages. First, it treats

the MDCP as a side-constrained minimum cost network flow problem where only a new set of flow distillation constraints (i.e. equations (13)) are added in addition to the conventional flow balance and bound constraints; Second, the basic graph corresponding to the basis is connected. Furthermore, the connectivity property of the basic graph is helpful in the development of our network simplex algorithm.

2.2. **Basic feasible graph.** In a conventional minimum cost network flow problem, a basis corresponds to a spanning tree so that the network simplex algorithm can easily operate from one spanning tree to another. Here in MDCP, the basis corresponding to a basic feasible flow $x$ constitutes a subgraph $G_B(x)$ composed by a spanning tree and some distillation arcs. Since the flow balance constraints in MDCP are the same as the conventional minimum cost network flow problem, $G_B(x)$ at least contains a spanning tree of $n-1$ basic arcs derivable from equations (7) to (12). Suppose each D-node $i$ contains $q_i$ distillation arcs, $q = \sum_{i \in N_D} q_i$, and that there are a total of $p$ D-nodes. Since equation (12) can be derived from equation (13), we can remove equation (12) and then the rank of the constraints (7) to (13) equals to $n + q - p - 1$ since there are a total of $p$ D-nodes. The basic feasible graph for an MDCP has the following properties:

**Lemma 2.1.** *Let $x$ be a basic feasible solution of an MDCP and $G_B(x)$ be the basic feasible graph corresponding to $x$, where $|N| = n$, $|N_D| = p$, $|L(i)| = q_i$ for each $i \in N_D$ and $q = \sum_{i \in N_D} q_i$. The basic graph has the following properties:*
*(i) The number of basic arcs is $n + q - p - 1$.*
*(ii) Any cycle of $G_B(x)$ includes at least one D-node.*
*(iii) $G_B(x)$ is connected.*
*(iv) Each D-node $i$ is connected with $q_i$ or $q_i + 1$ basic arcs.*
*(v) After removing $q_i - 1$ basic arcs for each D-node $i$, $G_B(x)$ can be reduced to a spanning tree.*

*Proof.* (i) Trivial.
(ii) See [8].
(iii) Our MDCP contains the same flow balance constraints corresponding to a connected spanning tree, as in the conventional minimum cost flow problem. Thus the basic graph of our MDCP is connected.
(iv) See [8].
(v) The proof is modified from [8]. By (iii), we know that any cycle in $G_B(x)$ passes at least one D-node. Since each D-node $i$ is connected by at least $q_i$ arcs by (iv), and if we remove $q_i - 1$ basic arcs for each D-node $i$ (i.e., totally removing $q - p$ arcs) from $G_B(x)$, then the remaining basic graph contains $n - 1$ basic arcs and remains connected without any cycle which corresponds to a spanning tree. □

Although our Lemma 2.1 seems similar to the properties proposed by Fang and Qi [8], they are not identical in the following sense: First, we generalized their results to deal with a capacitated MDCP whereas their results are only applicable for an uncapacitated MDCP; Second, we suggest a more specific way (property (v) in Lemma 2.1) to describe the relationship between $G_B(x)$ and its induced spanning tree, which helps us to design the optimality conditions in Section 2.3 and our graphical network simplex algorithm in Section 3.

2.3. **Dual variables and optimality conditions.** Let $\pi_i$ and $\tilde{\pi}_i$ be dual variables associated with the flow balance constraint for each node $i \in N_{\hat{O}}$ (equation (7) to (11)) and $i \in N_D$ (equation (12)), respectively. Let $v_{ij}$ denote dual variable associated with the flow distillation constraint (equation (13)) for each distillation arc $(i, j)$. The constraints of the dual problem of an MDCP can be formulated as follows:

$$\pi_i - \pi_j \le c_{ij} \qquad \forall i, j \in N_{\hat{O}} \tag{15}$$

$$\tilde{\pi}_i - v_{ij} - \pi_j \le c_{ij} \qquad \forall i \in N_D, j \in N_{\hat{O}} \tag{16}$$

$$\pi_i - \tilde{\pi}_j + \sum_{l \in L(j)} k_{jl} v_{jl} \le c_{ij} \qquad \forall j \in N_D, i \in E(j) \tag{17}$$

For each distillation arc $(j, l)$ leaving D-node $j$, define $\rho_{jl} = \tilde{\pi}_j - v_{jl}$ and $\pi_j = \sum_{l \in L(j)} k_{jl} \rho_{jl}$ . Since $\sum_{l \in L(j)} k_{jl} = 1$, we will have $\tilde{\pi}_j - \sum_{l \in L(j)} k_{jl} v_{jl} = \sum_{l \in L(j)} k_{jl}(\tilde{\pi}_j - v_{jl}) = \sum_{l \in L(j)} k_{jl} \rho_{jl} = \pi_j$. Equation (15) to (17) can be rewritten as

$$\pi_i - \pi_j \le c_{ij} \qquad \forall i \in N_{\hat{O}}, j \in N \tag{18}$$

$$\rho_{ij} - \pi_j \le c_{ij} \qquad \forall i \in N_D, j \in N_{\hat{O}} \tag{19}$$

We apply the upper bound technique for linear programming to solve a capacitated MDCP. Specifically, when $x_{ij} = u_{ij}$ for an arc $(i, j)$, one may consider its orientation to be reversed. This also means that the check on its dual feasibility constraint has to be conducted conversely (e.g. replace the $\le$ with $\ge$ in equations (18) and (19)). The upper bound technique can also be implemented in the original model proposed by Fang and Qi [8], but their basic graph can not be guaranteed to be connected, which complicates the procedure to traverse along arcs for checking the dual feasibility. On the other hand, the basic graph is shown to be connected in our model by Lemma 2.1(iii).

Let $B$, $L$, and $U$ be the set of basic arcs, non-basic arcs at lower bound, and non-basic arcs at upper bound, respectively. Thus the set of all arcs $A = B \cup L \cup U$. The dual optimality conditions are then as follows:

1. For each arc $(i, j) \in B$,

$$\pi_i - \pi_j = c_{ij} \qquad \forall i \in N_{\hat{O}}, j \in N \tag{20}$$

$$\rho_{ij} - \pi_j = c_{ij} \qquad \forall i \in N_D, j \in N_{\hat{O}} \tag{21}$$

2. For each arc $(i, j) \in L$,

$$\pi_i - \pi_j \le c_{ij} \qquad \forall i \in N_{\hat{O}}, j \in N \tag{22}$$

$$\rho_{ij} - \pi_j \le c_{ij} \qquad \forall i \in N_D, j \in N_{\hat{O}} \tag{23}$$

3. For each arc $(i, j) \in U$,

$$\pi_i - \pi_j \ge c_{ij} \qquad \forall i \in N_{\hat{O}}, j \in N \tag{24}$$

$$\rho_{ij} - \pi_j \ge c_{ij} \qquad \forall i \in N_D, j \in N_{\hat{O}} \tag{25}$$

3. **Network Simplex algorithm.** The network simplex algorithm is a specialized simplex algorithm designed specifically for solving network-type linear programming problems. The conventional network simplex algorithm is designed for minimum cost network flow problem and exploits graphical operations in order to efficiently calculate the basic feasible solutions. However, it cannot deal with the distillation

constraints of an MDCP. On the other hand, the network simplex algorithm proposed by Fang and Qi [8] for an MDCP is not complete in the sense that many steps in their algorithm are only algebraic operations. To fully exploit the advantage of graphical operations, we propose a network simplex algorithm including technical details in such steps as to allow us to obtain the initial basic feasible solutions, to pivot flows along basic feasible graphs, and to update dual basic solutions for solving a capacitated MDCP. Without loss of generality, we assume that our MDCP always has a finite optimal solution and that the degeneracy is resolved using anti-cycling techniques. Our network simplex algorithm contains the following steps:

**Step 0:** Start with an initial basic feasible flow on a basic feasible graph $G_B(x)$.
**Step 1:** Calculate dual basic solutions for $G_B(x)$.
**Step 2:** Check the dual feasibility conditions (equation (22) to (25)) for each arc in $L \cup U$. If no arcs violate the optimality conditions, then the flow $x$ is optimal and the algorithm terminates.
Otherwise, select a violating arc $(k, l)$ as the entering arc to the basis and continue **Step 3**.
**Step 3:** Conduct flow pivoting operations and determine the leaving arc $(v, z)$.
**Step 4:** Update dual variables, and then return to **Step 2** for the next iteration.

Our algorithm exploits several novel basis partitioning techniques which decompose a basic graph into components so that arc flows as well as node potentials can be efficiently calculated and updated. Detailed operations for each step are explained in the next sections.

3.1. **Obtaining an initial basic feasible flow.** Let $M$ be a very large number. We present the following procedure based on the Big-M method to compute an initial basic feasible flow:

**Step 1:** For each $i \in N_S$, include arc $(s, i)$ into the basis with $x_{si} := 0$. Also include arc $(s, t)$ into the basis with $x_{st} := \sum_{i \in N_S} u_i - \sum_{i \in N_T} d_j$.
**Step 2:** For each $i \in N_{\widehat{O}}$, add artificial arcs $(s, i)$ to be a basic arc with $u_{si} := M$, $c_{si} := M$, and $x_{si} := 0$.
**Step 3:** For each $i \in N_T$, add artificial arcs $(s, i)$ to be a basic arc with $u_{si} := M$, $c_{si} := M$, and $x_{si} := d_i$.
**Step 4:** For each $i \in N_D$, include each distillation arc $(i, j)$ where $j \in L(i)$ to be a basic arc with $x_{ij} := 0$.

In particular, Step 1 through Step 3 identify $n - p - 1$ basic arcs, and Step 4 identifies another basic arcs. The flow assignments are feasible since all the supply and demand are satisfied. Furthermore, basic dual variables can be calculated by setting $\pi_s := 0$, $\pi_i := -c_{si}$ for each $i \in N_{\widehat{O}}$, $\rho_{ij} := \pi_j + c_{ij}$ for each distillation arc $(i, j)$, and $\pi_i := \sum_{j \in L(i)} k_{ij}\rho_{ij}$ for each $i \in N_D$. This procedure takes $O(n + q - p)$ time.

3.2. **Calculating basic dual solutions.** Fang and Qi [8] outlined this procedure without detailed graphical implementation. In the present paper we propose a basis partitioning technique that decomposes the basic graph into $p+1$ *basic components*, in which each basic component is a tree and contains at least one D-node or one distillation arc. We show that all dual variables on the same basic component can

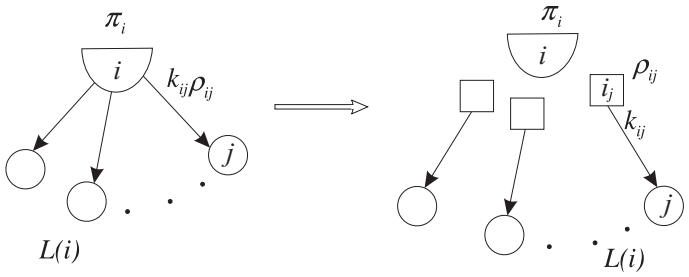



FIGURE 5. Detaching the distillation from a D-node

be expressed using a representative dual variable (e.g. the $\pi$ associated with some node in that basic component) due to its tree structure, so that we can use the $p+1$ representative dual variables to derive all the $n+q$ dual variables. A system of $p+1$ linear equations, composed by $\pi_i = \varsigma$ for some node $i \in N$ or $\rho_{\alpha\beta} = \varsigma$ for some distillation arc $(\alpha, \beta)$ with a constant $\varsigma$, as well as $p$ equations $\pi_i = \sum_{j \in L(i)} k_{ij}\rho_{ij}$ of for each $i \in N_D$, can be used to solve the $p+1$ representative dual variables, and then to derive all the $n+q$ dual variables.

3.2.1. *Decomposing the Basic Graph into Basic Components.* When solving basic dual variables for a minimum cost network flow problem, the network simplex algorithm starts from any node $i$, sets $\pi_i$ to be a fixed value $\varsigma$ (e.g. $\varsigma = 0$), and then calculates other dual variables by tracing basic arcs along the basic tree arcs. Here in MDCP, such a tracing operation is not trivial when a D-node is encountered. Specifically, when starting from a node $i \in N$ one may trace along basic arcs and calculate other dual variables $\pi$ or $\rho$ using equations (20) and (21). However, when a D-node $i$ is encountered, the tracing has to be stopped since the $q_i + 1$ dual variables in the equation $\pi_i = \sum_{j \in L(i)} k_{ij}\rho_{ij}$ associated with a D-node $i$ cannot be calculated using a single equation (i.e. equation (20) or (21)). A search algorithm such as the Depth-First Search (DFS) or Breadth-First-Search (BFS) can be used to trace the basic arcs and calculate dual variables using equation (20) and (21), as long as we do not crossover a D-node. Once the search algorithm backtracks to its starting node it can start from any unvisited node and conduct the same operations until all the nodes in $N$ have been visited. That is to say , D-nodes can be viewed as boundary nodes for the search algorithm.

To have a better illustration for this operation, we detach each distillation arc $(i, j)$ outgoing from each D-node $i$, and replace its tail node by a pseudo node $i_j$ referred to as a *side-node* (the squared nodes in Figure 5). A new dual variable $\pi_{i_j} = \rho_{ij}$ is assigned to the side-node $i_j$ for recording dual variable $\rho_{ij}$ associated with the distillation arc $(i, j)$. Thus equation (21) becomes $\pi_{i_j} - \pi_j = c_{ij}$ for each distillation basic arc $(i, j)$ and has a form similar to equation (20). Now we only need to consider dual variables $\pi$ associated with each node in $N$ and each side-node.

The detachment procedure disconnects each D-node $i$ and all the nodes it emanates to. By lemma 2.1(iv), we know that each D-node $i$ is connected by either $q_i$ or $q_i+1$ basic arcs. Lemma 2.1(v) also suggests that the disconnection of $q_i-1$ basic arcs for each D-node $i$ in $G_B(x)$ reduces the $G_B(x)$ to a spanning tree, denoted as $T(x)$. Therefore, the disconnection of $q_i$ basic arcs for each D-node $i$ in $G_B(x)$ will disconnect one more basic arc for each D-node $i$ in the spanning tree $T(x)$. Since

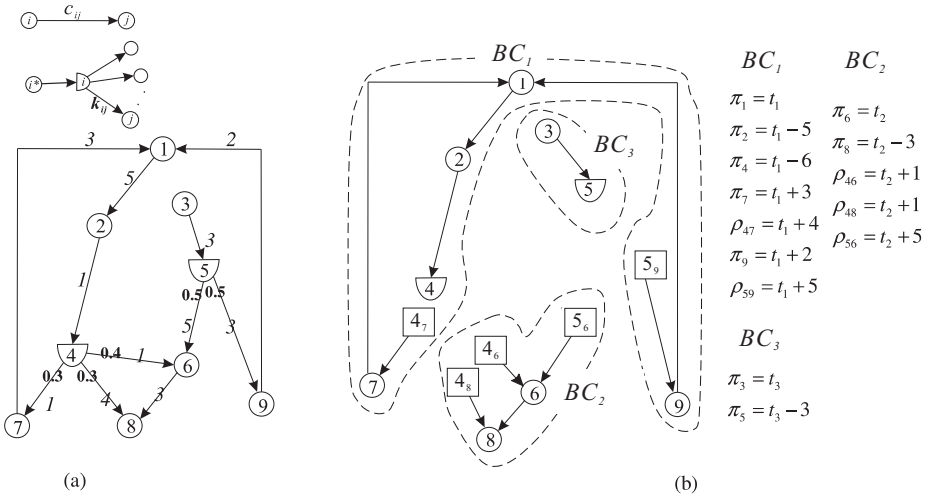



FIGURE 6. An example of basic components

there are a total of $p$ D-nodes, the detachment thus disconnects $p$ basic arcs from $T(x)$ and forms a forest of $p+1$ trees, where each tree is called a basic component. Note that each of these $p+1$ basic components contains at least one D-node or one side-node.

Since each basic component is a tree, one may arbitrarily choose a node inside a basic component and use its dual variable as a base to derive dual variables along the basic tree arcs for all the other nodes inside the same basic component. Specifically, inside each basic component, the search algorithm starts from a node, sets its dual variable as a base for that basic component, and traverses along basic arcs until it encounters a boundary node (i.e. a D-node or a side-node), then it stops and retreats to visit other unvisited nodes. Whenever a node is visited for the first time, its associated dual variable can be calculated by equation (20) or (21).

Figure 6(b) illustrates how a basic graph containing two D-nodes in Figure 6(a) is decomposed. In this example, three components ( $BC_1$, $BC_2$, and $BC_3$) can be identified, and three independent variables ( $\pi_1$, $\pi_6$, and $\pi_3$) can be used to derive all dual variables through the basic tree arcs.

For a D-node $i$ connecting with $q_i + 1$ basic arcs in $G_B(x)$, the detachment will group the D-node $i$ into the basic component containing its entering node $i^*$ (i.e. $i^* \in E(i)$), while all of its side-nodes belong to the other basic components. On the other hand, for a D-node $i$ connecting with $q_i$ basic arcs in $G_B(x)$, the detachment will create a basic component composed by an isolated D-node or side-node. For example, Figure 7(a) shows a case where the incoming arc $(i^*, i)$ for the D-node $i$ is non-basic, and Figure 7(b) gives another example where a distillation arc $(i, j)$ for the D-node $i$ is non-basic.

Each basic feasible graph can be uniquely decomposed using our basis partitioning technique. The following are some properties of the basic components.

**Lemma 3.1.** *(i) The number of basic components is $p+1$.*
*(ii) Each basic component is a tree.*
*(iii) The number of components containing at least one $\widehat{O}$-node (i.e. not an isolated*

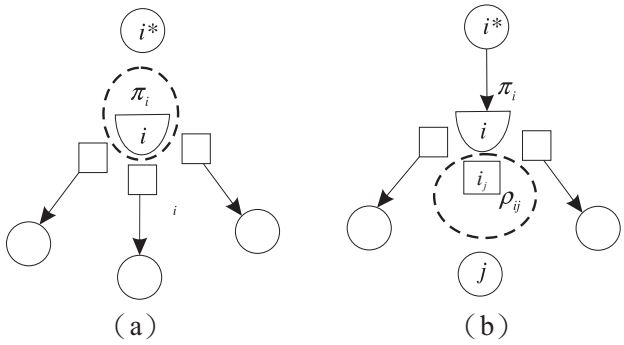



FIGURE 7. Examples of basic components composed by an isolated node

*D-node or side-node) is at most* $\min\{p+1, n-p\}$. *That is to say, the number of D-nodes whose adjacent arcs are all basic is at most* $\min\{p, n-p-1\}$.

*Proof.* (i) and (ii) have been explained in previous paragraphs. We give the proof for (iii) as follows:
(iii) If $p+1 \leq n-p$, we may distribute at least one $\widehat{O}$-node to each of these $p+1$ basic components, so that the maximum number of basic components containing at least one $\widehat{O}$-node is $p+1$. On the other hand, when $p+1 > n-p$, we may at most have $n-p$ among these $p+1$ basic components that contain at least one $\widehat{O}$-node. Therefore, there are at most $\min\{p+1, n-p\}$ basic components containing at least one $\widehat{O}$-node. Since these basic components must be formed by detaching the distillation arcs from those D-nodes whose adjacent arcs are all basic, there are at most $\min\{p, n-p-1\}$ such D-nodes. □

The following are the steps to decompose a basic graph into several basic components and assign a single variable to express each dual variable in the same component as follows:

**Step 0:** Given a basic feasible graph $G_B(x)$ corresponding to a basic feasible flow $x$. Construct an *augmented basic graph* $G_B^{'}(x)$ by first duplicating all the nodes and arcs from $G_B(x)$. Then, for each distillation arc $(i,j)$ in $G_B^{'}(x)$, detach its tail from the D-node $i$, and add a new side-node $i_j$ vas its new tail node whose dual variable $\pi_{i_j} = \rho_{ij}$.

**Step 1:** Initialize $p+1$ node sets ( $BC_i := \varnothing$, $i = 1, .., p+1$ ); unmark each node (i.e. each node in $N$ and each side-node) in $G_B^{'}(x)$; set $k = 1$.

**Step 2:** Select an unmarked node $r$ from $G_B^{'}(x)$, put it into $BC_k$, and set $\pi_r = t_k$

**Step 3:** Starting from node $r$, conduct a search algorithm to traverse along arcs in $G_B^{'}(x)$. When the search algorithm traverses from a marked node $i$ to an unmarked node $j$ along a basic arc $(i,j)$ (or $(j,i)$ ) in $G_B^{'}(x)$, we set $\pi_j = \pi_i - c_{ij}$ (or $\pi_j = \pi_i + c_{ij}$). When the search algorithm terminates, all dual variables in this component have been expressed by $t_k$. Set $k = k+1$;

**Step 4:** Repeat Step 2 and Step 3 until all the nodes in $G_B^{'}(x)$ are marked.

Note that the augmented basic graph $G_B^{'}(x)$ contains $n+q-p-1$ basic arcs and $n+q$ nodes. The search algorithm scans each arc and node exactly once in Step 3 and results in a total $\theta(n+q)$ time for this procedure.

3.2.2. *Solving a Smaller System of Linear Equations.* After decomposing the $p+1$ basic components, dual variable associated with each node inside the basic component has been expressed using the representative variable $t_i$. Thus, there are a total of $p+1$ representative dual variables. Similar to the conventional network simplex algorithm for the minimum cost flow problem where one can arbitrarily set a dual variable to a fixed value and then derive all other dual variables with respect to that dual variable via basic arcs, here we can also arbitrarily select a dual variable as a base and compute all other $p$ dual variables accordingly. For example, setting $\pi_1 = t_1 = 0$, we will have $p$ representative variables $(t_i : i = 2, ..., p+1)$ and a system of $p$ linear equations $(\pi_i = \sum_{j \in L(i)} k_{ij}\rho_{ij} : i \in N_D)$.

Take Figure 6 as an example. Using $\pi_1 = 0$ and the relations between dual variables as defined in Figure 6(b), the equations associated with flow distillation constraints: $\pi_4 = 0.4\rho_{46} + 0.3\rho_{47} + 0.3\rho_{48}$ and $\pi_5 = 0.5\rho_{56} + 0.5\rho_{59}$ can be expressed using $t_2$ and $t_3$, and we can compute $t_2 = -11.286$ and $t_3 = 0.5t_2 + 8 = 2.357$.

Although our basis partitioning technique requires us to solve a system of $p$ linear equations for calculating basic dual variables, this is already more efficient than solving a system of $n+q-p-1$ linear equations as required in the network simplex method proposed by Fang and Qi [8]. In fact, the efficiency of our method can be improved further, if a better ordering in the sequence of components to be solved can be identified. For example, in Figure 6(b), D-node 4 is the boundary node for two components ($BC_1$ and $BC_2$), and D-node 5 is the boundary node for three components ($BC_1$, $BC_2$, and $BC_3$). Setting $t_3 = 0$ will have to solve a $2 \times 2$ system of linear equations, whereas setting $t_1 = 0$ or $t_2 = 0$ will make the remaining system of linear equations become a triangular form, which can be solved more efficiently.

To speed up this procedure, we may reduce the number of linear equations required to be solved. We first identify the following three types of basic components: (1) a basic component composed of an isolated D-node or side-node (e.g. Figure 7), (2) a basic component that contains exactly one D-node, some $\widehat{O}$-nodes but no side-nodes (e.g. the $BC_3$ in Figure 6), and (3) a basic component that contains no D-node, some $\widehat{O}$-nodes, and some side-nodes whose associated distillation arcs are emanating from the same D-node in the adjacent basic component. We call these three types of basic components *leaf basic components* since their dual variables only depend on a single dual variable without interacting with others. Thus, dual variables inside a leaf basic component can first be left aside, and then later be derived from dual variables of other non-leaf basic components. By Lemma 3.1(iii), we know that there are at most $\min\{p+1, n-p\}$ basic components that are not leaf basic components. Thus, we may at most solve a system of $\min\{p+1, n-p\}$ linear equations which takes $O(\min\{p^3, (n-p)^3\})$ time. Then, the calculation on dual variables in the remaining leaf basic components takes $O(p+q)$ time. In summary, the procedure to calculate all dual variables takes $O(\min\{p^3, (n-p)^3\}+n+q)$ time.

3.3. **Finding an entering arc and pivoting flows.** After calculating basic dual variables, any arc in $L \cup U$ that violates the dual feasibility conditions (equation (22) to (25)) is eligible to enter the basis. A *pivoting graph*, obtained by adding the entering arc to the basic graph, contains more than one cycle since the original

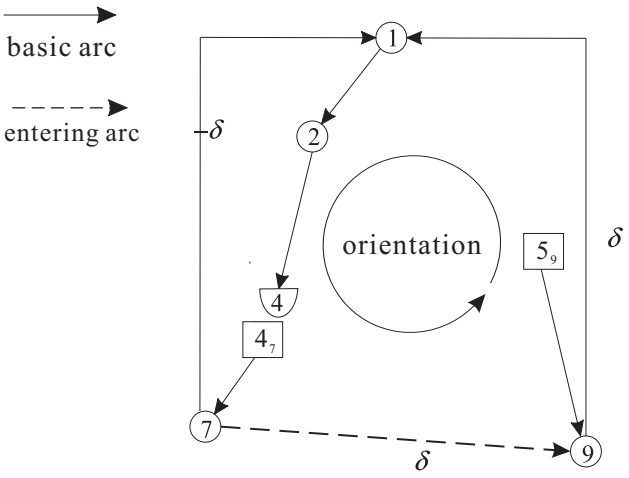



FIGURE 8. Pivoting flow inside a basic component

basic graph already contains cycles induced by the D-nodes. Thus the flow pivoting operations become more difficult and complicated. Similar to the computation on dual variables, the difficulty of flow pivoting lies in the design of an efficient procedure to pivot flows graphically, rather than algebraically. To this end, we apply the basis partitioning technique used in calculating dual variables to design efficient graphical flow pivoting operations. There are two types of entering arcs: either the entering arc connects two nodes of the same component or it does not. We provide different procedures to conduct flow pivoting for these two cases.

3.3.1. *The Entering Arc Connecting Nodes of the Same Basic Component.* Since each basic component is a tree, adding the non-basic entering arc $(k, l)$ in this case will induce a unique cycle inside the same basic component, as shown in Figure 8. In addition, nodes $k$, $l$, and other nodes on the induced cycle have to be $\widehat{O}$-nodes, since all the D-nodes will become leaf nodes in $G_B(x)$ (and thus D-nodes can not become internal nodes). The flow pivoting operations in this case are made up of the following steps:

**Step 1:** Add the non-basic entering arc $(k, l)$ into the $BC$ containing nodes $k$ and $l$.

**Step 2:** Identify the unique cycle induced by adding $(k, l)$ to the $BC$ containing nodes $k$ and $l$.

**Step 3:** If $(k, l) \in L$, we ship the maximum flow along the orientation of $(k, l)$ in the induced cycle, Update flows for the arcs in the induced cycle, and then determine the leaving arc $(v, z)$ that first achieves its flow bound.
Otherwise (i.e. $(k, l) \in U$), we ship the maximum flow along the opposite orientation of $(k, l)$ in the induced cycle, update flows for the arcs in the induced cycle, and then determine the leaving arc $(v, z)$ that first achieves its flow bound.

This procedure takes $O(n + q)$ time.

3.3.2. *The Entering Arc Connecting Nodes of Different Basic Components.* Three types of end nodes for the entering arc $(k,l)$ are possible in this case: (1) $k \in N_{\widehat{O}}, l \in N_{\widehat{O}}$ (2) $k \in N_{\widehat{O}}, l \in N_D$, and (3) $k \in N_D, l \in N_{\widehat{O}}$. In general, the entering arc merges two components and reduces the number of basic components from $p+1$ to $p$. Note that in this case the entering arc induces no cycle in the newly merged component, and thus we have to design a new graphical procedure to pivot flows. Here we will exploit the basis partitioning techniques used for calculating basic dual variables in Section 3.2 to compute flows in the pivoting process.

To speed up the calculation, we conduct a graph compacting process to remove those nodes not eligible to ship flows in the pivoting graph, as well as their associated arcs. In particular, two types of nodes are removable: (1) any $\widehat{O}$-node connecting with one arc, and (2) any node inside a leaf basic component. This compacting procedure may be repeated until all the $\widehat{O}$-nodes are connected with at least two basic arcs and all the leaf basic components are removed. Since each compacting operation removes at least one node and one basic arc, it takes a total of $O(n+q)$ time.

The graph compacting procedure reduces the number of components in a pivoting graph from $p$ to $\tilde{p}$. $\widetilde{G} = (\widetilde{N}, \widetilde{A})$ denotes the remaining pivoting graph after the compacting process. We give four properties for $\widetilde{G}$ as follows:

**Lemma 3.2.** *(i) Any leaf node in $\widetilde{N}$ is either a D-node or a side node.*
*(ii) Any D-node $i$ in $\widetilde{N}$ has to be adjacent to $q_i + 1$ arcs.*
*(iii) $\tilde{p} \le \min\{p, n-p\}$.*
*(iv) $\widetilde{G}$ contains $\tilde{p}$ D-nodes.*

*Proof.* (i) and (ii) are trivial since the compacting process removes all the leaf $\widehat{O}$-nodes, as well as those isolated D-nodes or side-nodes. We give the proof for (iii) and (iv) as follows:
(iii) If $p+1 \le n-p$, then we know that $p < n-p$ and thus $\tilde{p} \le p$ since the $n-p$ $\widehat{O}$-nodes can at most cover all the $p$ components in the pivoting graph. On the other hand, if $p+1 > n-p$, then we know that $p \ge n-p$ and thus $\tilde{p} \le n-p$ since the $n-p$ $\widehat{O}$-nodes can at most cover $n-p$ of the $p$ components in the pivoting graph. Therefore, $\tilde{p} \le \min\{p, n-p\}$.
(iv) The $\tilde{p}$ components have to include the merged basic components induced by the entering arc $(k,l)$, since the entering arc is the source of any flow change. This means that there were $\tilde{p}+1$ basic components if we exclude the entering arc from $\widetilde{A}$. Since these $\tilde{p}+1$ basic components must be formed by detaching the distillation arcs from the $\tilde{p}$ D-nodes, and these D-nodes will not be removed by (ii), so we know $\widetilde{N}$ contains $\tilde{p}$ D-nodes. □

To conduct the flow pivoting operation, we have to identify the relationship of flow changes on basic arcs with respect to the flow change on the entering arc. To this end, we first set $\Delta_{kl} = 1$ (or $\Delta_{kl} = -1$) to represent the shipping of one unit of flow along the entering arc $(k,l)$ if $(k,l) \in L$ (or $(k,l) \in U$), and then for each D-node $i$ we assign the flow change along its incoming arc to be $\Delta_{i^*i}$ (thus we have in total $\tilde{p}$ variables of $\Delta_{i^*i}$ by Lemma 3.2(iii)). Using the flow distillation and balance constraints, we can derive the flow change on each arc in $\widetilde{A}$ in terms of $f_{i^*i}$ for each D-node $i$. Specifically, the flow in each distillation arc can be derived by

$\Delta_{ij} = k_{ij}\Delta_{i^*i}$ for each D-node $i$. Since each component in $\widetilde{G}$ is a tree with leaf nodes as D-nodes or side-nodes, all the flow change entering or leaving each leaf node can be expressed by $\Delta_{i^*i}$ for each D-node $i$.

In addition, using the flow balance constraints associated with each internal node, we can conduct a search algorithm inside a component to traverse from leaf nodes to all other internal nodes and derive the flow change entering or leaving each internal node in terms of $\Delta_{i^*i}$ for each D-node $i$. Thus, for each component in $\widetilde{G}$, we may arbitrarily select an internal node as its root node and use the flow balance equation associated with each root node to construct a system of $\tilde{p}$ linear equations. Note that this system of $\tilde{p}$ linear equations has a rank equal to $\tilde{p}-1$ since the flow balance constraints have to be satisfied for each node and for the entire system itself. Thus, one of the $\tilde{p}$ flow balance equations can be removed. On the other hand, by considering the additional equation $\Delta_{kl} = 1$ (or $\Delta_{kl} = -1$, depending on whether $(k,l) \in L$ or $(k,l) \in U$) representing the unit flow change for the entering arc $(k,l)$, a system of $\tilde{p}$ linear equations can be constructed to solve the $\tilde{p}$ variables of $\Delta_{i^*i}$ for each D-node $i$, and then all the relative flow changes on each arc in $\widetilde{A}$ can be derived with respect to $\Delta_{kl}$.

Obtaining the relative flow change for each arc in $\widetilde{A}$, we can conduct a minimum ratio test to calculate $\theta_{vz} = \min_{(i,j)\in\widetilde{A},\Delta_{ij}\neq 0}\{(u_{ij}-x_{ij})/\Delta_{ij} : \Delta_{ij} > 0, -x_{ij}/\Delta_{ij} : \Delta_{ij} < 0\}$ for a leaving arc $(v,z)$, update flows by $x_{ij} = x_{ij} + \theta_{vz}\Delta_{ij}$ for each arc $(i,j)$ in $\widehat{A}$ with $\Delta_{ij} \neq 0$, and then remove $(v,z)$ to form another basic graph for the next iteration.

Take the pivoting graph in Figure 9(a) for example. By adding the entering arc it merges two basic components $BC_1$ and $BC_3$. Setting the amount of flow changes on arcs and to be $a$ and $b$, respectively, we may derive the flow changes entering or leaving the leaf nodes, as shown in Figure 9(b). Selecting nodes 1 and 6 to be the root nodes in $BC_1$ and $BC_2$, respectively, we can derive all the flow changes entering or leaving each internal node as shown in Figure 9(c). Two flow balance equations, $a+b-0.3a-0.5b = 0$ and $-0.3a-0.4a-0.5b = 0$, associated with these root nodes can be formulated. Since these two equations depend on each other, we replace the second equation by $b = 1$ since the entering arc $(1,3) \in L$. Therefore we can derive the flow change for each arc in the pivoting graph relative to the flow change on the entering arc, as shown in Figure 9(d).

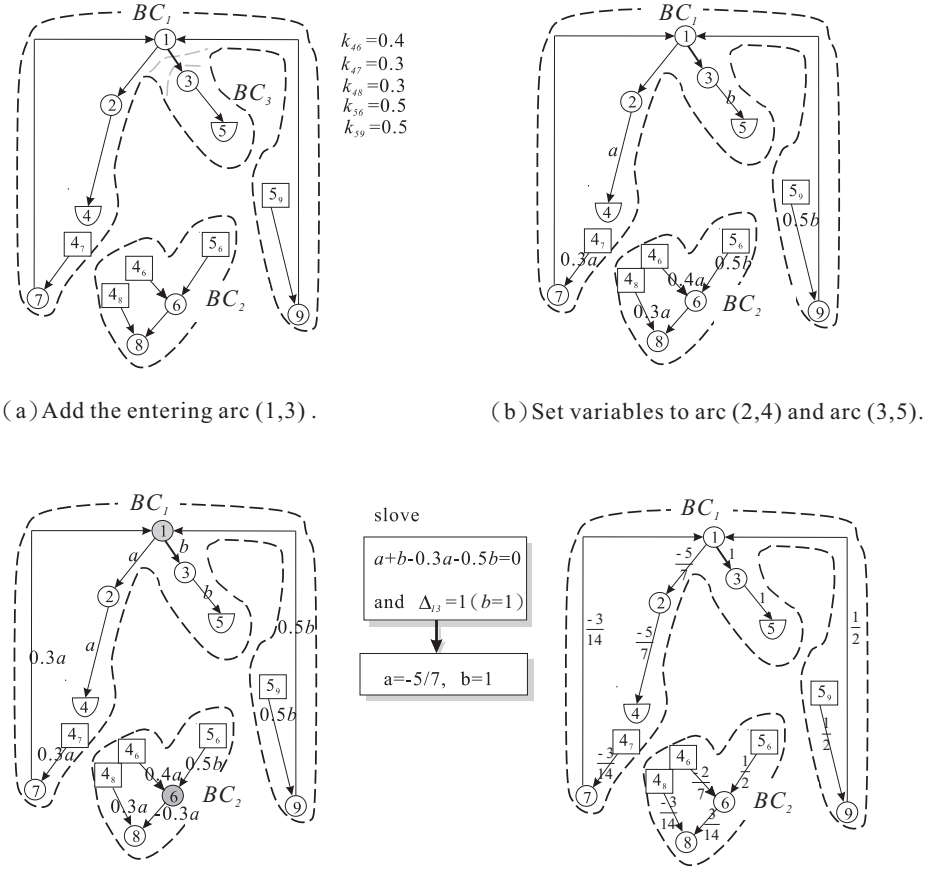
The procedure to pivot the flow in the case where the entering arc connects nodes of different basic components is described as follows:

**Step 1:** Add the non-basic entering arc $(k,l)$ into $G_B(x)$ to construct a pivoting graph.

**Step 2:** Conduct the graph compacting procedure which repeatedly removes any $\widehat{O}$-node connecting with one arc and any node inside a leaf basic component, as well as their associated arcs in a pivoting graph, until no more such node exists. We call the compacted graph as $\widetilde{G} = (\widetilde{N}, \widetilde{A})$.

**Step 3:** For each node $i \in N_D$ in $\widetilde{N}$, we assign a flow change variable $\Delta_{i^*i}$ for its incoming arc $(i^*, i)$, and then calculate $\Delta_{ij} = k_{ij}\Delta_{i^*i}$ for each distillation arc $(i,j)$.

**Step 4:** For each component in $\widetilde{G}$, we select some $\widehat{O}$-node in this component as its root node, conduct a search algorithm to traverse from leaf nodes to the root




(a) Add the entering arc (1,3).

(b) Set variables to arc (2,4) and arc (3,5).

(c) Calculate all $\Delta_{ij}$ by variables $a$ and $b$ (node 1 and 6 are root nodes).

(d) Get all $\Delta_{ij}$ by solving the flow balance constraint of node 1 and $\Delta_{13}$=1.

FIGURE 9. Computing the relative flow changes in a pivoting graph

node, and then derive the flow change on any arc inside that component using the flow balance constraints associated with each internal node.

**Step 5:** If $(k,l) \in L$, we solve the system of equations composed by $\Delta_{kl} = 1$ and the flow balance constraints associated with $\tilde{p}-1$ root nodes in $\widetilde{N}$ to obtain $\Delta_{ij}$ for each $(i,j) \in \widetilde{A}$.
Otherwise (i.e. $(k,l) \in U$), we solve the system of equations composed by $\Delta_{kl} = -1$ and the flow balance constraints associated with $\tilde{p}-1$ root nodes in $\widetilde{N}$ to obtain $\Delta_{ij}$ for each $(i,j) \in \widetilde{A}$.

**Step 6:** Calculate $\theta_{vz} = \min_{(i,j)\in\tilde{A},\Delta_{ij}\neq 0}\{(u_{ij}-x_{ij})/\Delta_{ij} : \Delta_{ij} > 0, -x_{ij}/\Delta_{ij} : \Delta_{ij} < 0\}$ for a leaving arc $(v,z)$.

**Step 7:** For each arc $(i,j) \in \widetilde{A}$ with $\Delta_{ij} \neq 0$, we update its flow by $x_{ij} := x_{ij} + \theta_{vz}\Delta_{ij}$.

The compacting procedure takes $O(n+q-p)$ time to construct $\widetilde{G}$. Setting the flow change variables for arcs connecting with D-nodes takes $O(\tilde{p}+\tilde{q})$ time, where

$\tilde{q}$ represents the total number of distillation arcs in $\widetilde{A}$. The search algorithm takes $O(\tilde{p}(|\widetilde{A}|-\tilde{p}-\tilde{q}))$ time to derive the flow changes on the other arcs in $\widetilde{A}$, since there are $|\widetilde{A}|-\tilde{p}-\tilde{q}$ arcs in $\widetilde{A}$ with both end nodes to be $\widehat{O}$-nodes, and each flow change has to be expressed using the $\tilde{p}$ variables. Solving the system of $\tilde{p}$ equations takes $O(\tilde{p}^3)$ time. The min-ratio test and flow updating operations for each arc in $\widetilde{A}$ take $O(|\widetilde{A}|)$ time. Thus the procedure to calculate flow changes in the pivoting graph can be done in $O(\tilde{p}^3+\tilde{p}|\widetilde{A}|+n+q) \leq O(\min\{p^3+np,(n-p)^3+n(n-p)\}+q)$ time, since $\tilde{p} \leq \min\{p, n-p\}$ by Lemma 3.2(iii).

Different from our basis partitioning technique, Sheu et al. [19] proposed a different partitioning technique to calculate the flow change on each arc in a given augmenting subgraph for solving a maximum flow problem in a distribution network. After constructing the compacted pivoting graph $\widetilde{G}=(\widetilde{N},\widetilde{A})$, we can use $\breve{p}$ $\widehat{O}$-nodes connected with more than two arcs as the dividing nodes to divide $\widetilde{G}$ into $\breve{p}+1$ *compatible components*, where the flow change on each arc inside a component can be expressed by a single variable. After solving the system of $\breve{p}+1$ equations composed by the flow balance equations associated with $\breve{p}$ dividing nodes and $\Delta_{kl}=1$ (or $\Delta_{kl}=-1$, depending on whether $(k,l)\in L$ or $(k,l)\in U$ ), the flow change on each arc in $\widetilde{A}$ can be expressed by $\Delta_{kl}$. Since there are at most $\breve{p}$ such $\widehat{O}$-nodes and $\breve{p}=O(n-p)$, their technique takes $O((n-p)^3)$ time. Therefore, our technique is at least asymptotically similar to theirs for the cases with more D-nodes. For the cases with fewer D-nodes than $\widehat{O}$-nodes, our technique is asymptotically faster than theirs.

3.4. **Updating dual variables.** After pivoting the flows and removing the leaving arc, the network simplex algorithm updates dual variables for the new basic graph. The new dual variables associated with the updated basic graph can be calculated from scratch using the procedure proposed in Section 3.2. However, since the basic graphs of two successive pivoting iterations share many common components, a more efficient dual variable update scheme can take advantage of the basic graph of the previous iteration. In this paper we propose a dual variables update procedure that exploits the structure of the basic components in the previous procedure and saves more computational efforts than the procedure in Section 3.2.

Let $bc(i)$ and $bc(i_j)$ represent the index of the basic component containing the node $i\in N$ and the side-node $i_j$, respectively. Suppose $G_B(x)$ is a basic graph in some network simplex iteration before flow pivoting, and that the flow pivoting procedure selects $(k,l)$ and $(v,z)$ to be the entering and leaving arc, respectively. Let $g=bc(k)$, $h=bc(l)$, and $w=bc(z)$. If we remove the leaving basic arc $(v,z)$ from $G_B(x)$, $BC_w$ will be split into two basic components: $BC_{bc(z)}$ and $BC_{bc(v)}$. For the sake of convenience, let $bc(z)=w$ and $bc(v)=p+2$, and there are a total of $p+2$ basic components in the pivoting graph, excluding the entering and leaving arcs. Since all the arcs in these $p+2$ basic components remain basic, dual variables still satisfy equations (20) and (21). Thus, dual variables inside each of these $p+2$ basic components will have the same amount of change, but different basic component may have a different amount of change. Let $\delta_\alpha$ record the change for each dual variable inside $BC_\alpha$ for $\alpha=1,...,p+2$. Let $\pi_i^{old}$, $\rho_{ij}^{old}$ and $\pi_i^{new}$, $\rho_{ij}^{new}$ denote dual variables associated with a node $i\in N$ and a distillation arc $(i,j)\in A$ in two successive network simplex iterations, respectively. Then, $\pi_i^{new}:=\pi_i^{old}+\delta_{bc(i)}$and $\rho_{ij}^{new}:=\rho_{ij}^{old}+\delta_{bc(i_j)}$.

Adding the non-basic entering arc $(k,l)$ to the $p+2$ basic components merges $BC_g$ and $BC_h$ into a new and larger component $BC_g \cup BC_h$ in the pivoting graph. For the sake of convenience, let $BC_g := BC_g \cup BC_h$and $BC_h := BC_{p+2}$. This merger integrates the variables of two components and reduces the number of components and variables by one. Now we only need to consider $p+1$ components (i.e. $BC_\alpha$ for $\alpha = 1, 2, ..., p+1$). Depending on the type of node $k$, $\delta_h$ can be expressed by $\delta_g$ using one of the following two equations:

1. If node $k$ is an $\widehat{O}$-node, it satisfies:

$$\begin{aligned} & \pi_k^{new} - \pi_l^{new} = c_{kl} \\ \Longrightarrow & \pi_k^{old} + \delta_g - \pi_l^{old} - \delta_h = c_{kl} \\ \Longrightarrow & \delta_h = \delta_g - c_{kl} - \pi_l^{old} + \pi_k^{old} \end{aligned} \tag{26}$$

2. If node $k$ is a D-node, it satisfies:

$$\begin{aligned} & \rho_{kl}^{new} - \pi_l^{new} = c_{kl} \\ \Longrightarrow & \rho_{kl}^{old} + \delta_g - \pi_l^{old} - \delta_h = c_{kl} \\ \Longrightarrow & \delta_h = \delta_g - c_{kl} - \pi_l^{old} + \rho_{kl}^{old} \end{aligned} \tag{27}$$

Now we can form a system of $p+1$ equations composed by $\delta_g = 0$, as well as the $p$ equations of $\delta_{bc(i)} = \sum_{j \in L(i)} k_{ij} \delta_{bc(i_j)}$ for each D-node $i$. Moreover, each equation can be expressed using $p+1$ variables composed $\delta_\alpha$ by for $\alpha = 1, 2, ..., p+1$. Therefore, the amount of change for each dual variable can be calculated by solving this system of $p+1$ linear equations.

Similar to the procedure in Section 3.2, we may speed up the procedure by updating dual variables for those non-leaf basic components first, and then later for the leaf basic components. The procedure to update dual variables is as follows:

**Step 0:** Initialize $H := \varnothing$, $N_{D1} := \varnothing$, where $H$ stores the indices of leaf basic components, and $N_{D1}$ stores the indices of the D-node associated with leaf basic components.
Let $\pi_i^{old}$ and $\rho_{ij}^{old}$ denote the original dual variable associated with a node $i \in N$ and a distillation arc $(i,j) \in A$, respectively.
For each node $i$ and side-node $i_j$, set $bc(i)$ and $bc(i_j)$ to be the index of the basic component that contains $i$ and $i_j$, respectively.

**Step 1:** Remove the leaving arc $(v,z)$ and split $BC_{bc(z)}$ into $BC_{bc(z)}$ and $BC_{bc(v)}$. For the sake of convenience, let $w := bc(z)$ and $bc(v) := p+2$.
For each basic component $\alpha = 1, ..., p+2$, let $\delta_\alpha$ represent the amount of change for each dual variable inside $BC_\alpha$.
For the entering arc $(k,l)$, let $g := bc(k)$ and $h := bc(l)$.

**Step 2:** Calculate $\delta_h$ by $\delta_g$ using $\delta_h = \delta_g - \pi_l^{old} + \pi_k^{old} - c_{kl}$ or $\delta_h = \delta_g - \pi_l^{old} + \rho_{kl}^{old} - c_{kl}$, depending on whether node $k$ is an $\widehat{O}$-node or a D-node.

**Step 3:** Add the entering arc $(k,l)$ to merge $BC_h$ into $BC_g$. For the sake of convenience, let $BC_g := BC_g \cup BC_h$ and $BC_h := BC_{p+2}$.

**Step 4:** Identify the leaf basic components, add their indices into $H$, and add their associated D-nodes into $N_{D1}$.

**Step 5:** Set $\delta_r = 0$ for some $BC_r$ not in $H$.

**Step 6:** Solve the system of equations composed by $\delta_{bc(i)} = \sum_{j \in L(i)} k_{ij} \delta_{bc(i_j)}$ : $\forall i \in N_D \backslash N_{D1}$ and $\delta_r = 0$.

**Step 7:** Solve the remaining $\delta_k : k \in H$ by $\delta_{bc(i)} = \sum_{j \in L(i)} k_{ij}\delta_{bc(i_j)} : \forall i \in N_{D1}$.

**Step 8:** Update each dual variable by $\pi_i^{new} = \pi_i^{old} + \delta_\alpha$ or $\rho_{jk}^{new} = \rho_{jk}^{old} + \delta_\alpha$for each node $i$ and side-node $j_k$.

Although this procedure is practically faster than the procedure in Section 3.2, it has the same theoretical complexity as $O(\min\{p^3, (n-p)^3\} + n + p)$.

4. **Conclusions.** To efficiently solve the minimum distribution cost problem proposed by Fang and Qi [8], this paper provided the first detailed graphical procedures to implement the network simplex algorithm. We have carefully designed a basis portioning technique to decompose a basic graph into several basic components divided by D-nodes, where each component corresponds to a tree. We also provided sound theoretical background to support our algorithm, and proposed a set of graphical operations to efficiently conduct the calculation for both primal and dual variables. With our techniques, many computational efforts for calculating the basic variables can be reduced, compared to other methods in the literature. Techniques to speed up our algorithm have also been investigated with theoretical support. Although omitted in this paper, for future research, we suggest investigating the degeneracy issues, which are usually encountered in simplex-like algorithms.

**Acknowledgments.** We would like to thank referee and editor for their valuable comments. I-Lin Wang was partially supported by the National Science Council of Taiwan under Grant NSC95-2221-E-006-268.

## REFERENCES

[1] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, "Network Flows: Theory, Algorithms and Applications," Prentice Hall, Englewood Cliffs, New Jersey, U.S.A., 1993.

[2] R. Barr, F. Glover and D. Klingman, *Enhancements to spanning tree labeling procedures for network optimization*, INFOR, **17** (1979), 16–34.

[3] G. H. Bradley, G. G. Brown and G. W. Graves, *Design and implementation of large scale primal transshipment algorithms*, Management Science, **24** (1977), 1–34.

[4] M. D. Chang, C. H. J. Chen and M. Engquist, *An improved primal simplex variant for pure processing networks*, ACM Transactions on Mathematical Software, **15** (1989), 64–78.

[5] C. H. J. Chen and M. Engquist, *A primal simplex approach to pure processing networks*, Management Science, **32** (1986), 1582–1598.

[6] E. Cohen and N. Megiddo, *Algorithms and complexity analysis for some flow problems*, Algorithmica, **11** (1994), 320–340.

[7] E. W. Dijkstra, *A note on two problems in connexion with graphs*, Numerische Mathematik, **1** (1959), 269–271.

[8] S. C. Fang and L. Qi, *Manufacturing network flows: A generalized network flow model for manufacturing process modeling*, Optimization Methods and Software, **18** (2003), 143–165.

[9] L. R. Ford and D. R. Fulkerson, *Maximal flow through a network*, Canadian Journal of Mathematics, **8** (1956), 399–404.

[10] F. Glover, D. Karney and D. Klingman, *Implementation and computational comparisons of primal, dual and primal-dual computer codes for minimum cost network flow problems*, Networks, **4** (1974), 191–212.

[11] F. Glover, D. Karney, D. Klingman and A. Napier, *A computation study on start procedures, basis change criteria and solution algorithms for transportation problems*, Management Science, **20** (1974), 793–813.

[12] F. Glover, D. Klingman and J. Stutz, *Augmented threaded index method for network optimization*, INFOR, **12** (1974), 293–298.

[13] E. L. Johnson, *Networks and basic solutions*, Operations Research, **14** (1966), 619–623.

[14] J. L. Kennington and R. V. Helgason, "Algorithms for Network Programming," John Wiley & Sons, New York, NY, USA, 1980.

[15] J. L. Kennington and R. V. Helgason, *Minimum cost network flow algorithms*, In "M.G.C. Resende and P.M. Pardalos, editors, Handbook of Optimization in Telecommunications," chapter 6, 147–162. Springer, 2006.
[16] J. Koene, "Minimal Cost Flow in Processing Networks, A Primal Approach," PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 1982.
[17] H. Lu, E. Yao and L. Qi, *Some further results on minimum distribution cost flow problems*, Journal of Combinatorial Optimization, **11** (2006), 351–371.
[18] J. Mo, L. Qi and Z. Wei, *A network simplex algorithm for simple manufacturing network model*, Journal of Industrial and Management Optimization, **1** (2005), 251–273.
[19] R. L. Sheu, M. J. Ting and I. L. Wang, *Maximum flow problem in the distribution network*, Journal of Industrial and Management Optimization, **2** (2006), 237–254.
[20] P. Venkateshan, K. Mathur and R. H. Ballou, *An efficient generalized network-simplex-based algorithm for manufacturing network flows*, Journal of Combinatorial Optimization, **15** (2008), 315–341.
[21] I. L. Wang and J. C. Lin, *Solving maximum flows on distribution networks: Network compaction and algorithm*, In "Proceedings of the 10th Annual Conference of the Asia-Pacific Decision Science Institute (APDSI)," Taipei, Taiwan, June 2005.
[22] I. L. Wang and Y. H. Yang, *On solving the uncapacitated minimum cost flow problems in a distribution network*, International Journal of Reliability and Quality Performance, **1** (2008), 53–63.

Received October 2008; 1st revision June 2009; final revision July 2009.

*E-mail address*: ilinwang@mail.ncku.edu.tw
*E-mail address*: shioujielin@gmail.com