JOURNAL OF INDUSTRIAL AND
MANAGEMENT OPTIMIZATION
Volume **12**, Number **1**, January **2016**

doi:10.3934/jimo.2016.12.117

pp. **117–140**

A COMPACTION SCHEME AND GENERATOR FOR DISTRIBUTION NETWORKS

I-Lin Wang and Ju-Chun Lin

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

Abstract. In a distribution network, materials or products that go through a decomposition process can be considered as flows entering a specialized node, called D-node, which distributes each decomposed flow along an outgoing arc. Flows on each arc emanating from a D-node have to obey a pre-specified proportional relationship, in addition to the capacity constraints. The solution procedures for calculating optimal flows over distribution networks in literature often assumes D-nodes to be disjoint, whereas in reality D-nodes may often connect to each other and complicate the problem. In this paper, we propose a polynomial-time network compaction scheme that compresses a distribution network into an equivalent one of smaller size, which can then be directly solved by conventional solution methods in related literature. In order to provide test cases of distribution networks containing D-nodes for computational tests in related research, we implement a random network generator that produces a connected and acyclic distribution network in a compact form. Mathematical properties together with their proofs are also discussed to provide more insights in the design of our generator.

1. **Introduction.** In a supply chain, the materials, semi-products or products in each stage are often processed and distributed into several channels. For example, suppose one unit of product A is made of two units of material B and one unit of material C. The relationship between materials and a product can be illustrated as a *product tree* (see Figure 1) suggested by Kung and Chern [12]. Material B and C may be purchased from different vendors by different channels. There may be several manufacturers who make product A with different production rates and obtain materials B and C via different channels.

We may integrate product trees with the supply chain network. Using Figure 1(a) and Figure 2 as examples, there are two manufactures (M_1 and M_2) and three vendors (V_1, V_2 and V_3) denoted by white circles. The product tree in a supply chain network can be represented by gray circles and half-circles which specify the process of decomposing products or semi-products into materials. The materials or semi-products are then purchasable from several vendors. For each outgoing arc of a gray half-circle, we associate a bracket with a number inside indicating the quantities of materials required for composing one unit of product (referring to Figure 1(a)). An arc connecting circles or half-circles may represent a process of production, decomposing, shipping, or outsourcing. We also associate a parenthesis for each arc which contains a letter in the left that indicates the object (i.e. material,

2010 *Mathematics Subject Classification.* Primary: 90B10, 05C85; Secondary: 05C21.

Key words and phrases. Distribution network, proportional flow constraint, compaction scheme, network generator, network optimization.

The reviewing process of the paper was handled by Yi-Kuei Lin as a Guest Editor.

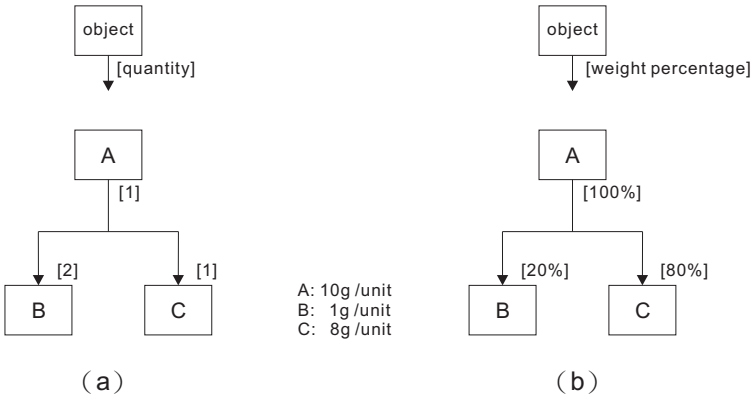

FIGURE 1. A product tree example

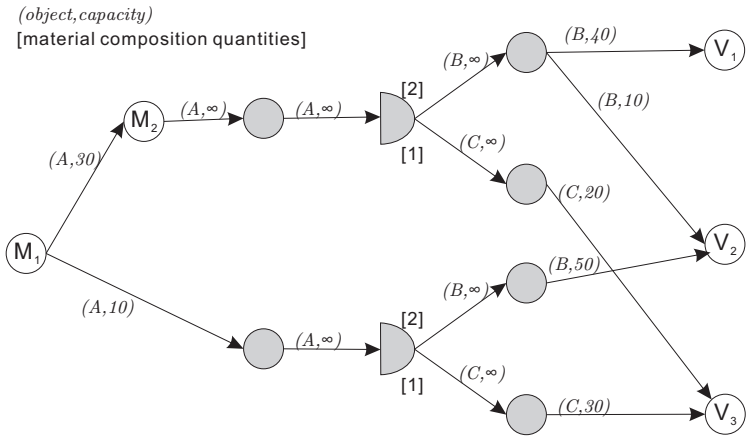

FIGURE 2. A supply chain example

semi-product, or product) to be processed and a number in the right representing the capacity (i.e. maximal number of objects processed) for the process. If a process has unlimited capacity, we use a "∞" to represent its capacity.

The scenario in Figure 2 is described as follows: Both M_1 and M_2 produce product A. The manager in M_1 will consider two strategies to fulfill the order: one is to produce products by his factory M_1, and the other is to outsource to M_2. One unit of A is made of two units of B and one unit of C. M_2 can purchase B from either V_1 or V_2, while M_1 can purchase B only from V_2.

In general, manufacturers either produce final products made of materials purchased from several vendors, or put part of their production out to contract with other manufacturers. Thus a final product may be made through several channels in different stages of a supply chain. Identifying the bottleneck (i.e. the set of channels that achieve their capacity and thus could not be improved) for a supply chain becomes an important issue to improve the entire efficiency. In other words, calculating the maximum throughput (i.e. product flow) for either the entire chain or part of the chain helps us to evaluate the chain's efficiency. However, the conventional maximum flow algorithms can not be directly applied to this supply chain

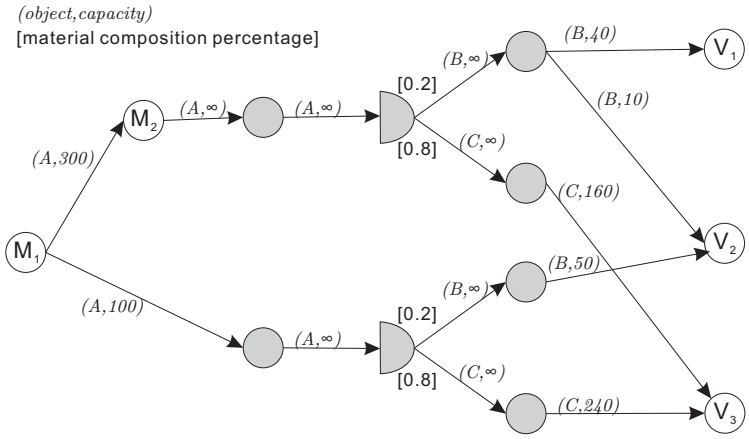

FIGURE 3. A supply chain example with unified units

since this problem differs from the conventional one by two points: first, units of flow are not unified in different arcs for this problem; and second, the amount of flow enters a gray half-circle has to be distributed by the predefined proportion.

To unify the flow units, we may use the weight instead of the quantities for different objects. In particular, the product tree in Figure 1(a) can be transformed into the product tree in Figure 1(b) since 10g of A (1 unit) can be made of 2g of B (2 units) and 8g of C (1 unit). Furthermore, we may integrate the distributed percentage of flow as shown in Figure 1(b) to transform Figure 2 into Figure 3 where the flow units are unified and the distributed flow percentage gives more intuitive illustration. For example, the capacity of arc (M_1,M_2) means M_2 can provide at most 300g of product A to M_1. To produce x g of product A, M_2 has to purchase $0.2x$ g of material B and $0.8x$ g of material C. Also, M_2 can purchase at most 40g and 10g of B from V_1 and V_2, respectively. This transformed network thus only differs from the conventional one by the appearance of gray half-circles and the flow distribution constraints associated with their leaving arcs. Such a network model is called a *distribution network*, a special case of a manufacturing network flow model investigated by Fang and Qi [8]. In their model, a gray half-circle is called a *distillation* node, or a D-node in short, and flows passing through a D-node has to be distributed according pre-defined ratios (e.g. 20% and 80%) to its outgoing arcs. In particular, 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 enforced proportional flow constraints associated with a D-node, in addition to the conventional flow balance and capacity constraints (see [1]), makes the calculation of optimal flows over distribution networks more complicated.

The distribution networks appear often in real-world applications. For example, as pointed out by Lyon et al. [15] and Denton et al. [7], in semiconductor manufacturing, the so-called *binning* or sorting process refers to assigning parts specific identities only after testing them. After the wafer is completed and tested, depending on the qualities of products, there may be a 50%, 30%, and 20% chance that it will be classified as Device A, B, or C. Similarly, in an oil refinery, crude oil is processed and refined into more useful products such as petroleum naphtha, gasoline, diesel fuel, asphalt base, heating oil, kerosene and liquefied petroleum gas.

Similar to the manufacturing network proposed by Fang and Qi [8], the *pure processing network* that contains nodes of refining, transportation, and blending processes was firstly introduced by Koene [11]. Chen and Engquist [5], and Chang et al. [4] further investigated the minimum cost flow problems on pure processing networks, which was called a *minimum distribution cost problem* (MDCP) by Fang and Qi [8]. Recently, Lu et al. [14] investigated a generalized MDCP that considers distillation arcs with gains or losses (i.e. $\sum_{(i,j)} k_{ij}$ may not equal to 1). All of these works gave variants of algebraic primal simplex method that exploited the basis partitioning technique to cope with the additional proportional flow constraints.

Sheu et al. [16] proposed the first graphical *multi-labeling method* to solve the maximum flow problem on a distribution network by adopting the concept of the Depth-First Search (DFS) which tries out every augmenting subgraph that goes to the sink or source and satisfies flow distillation constraints. After finding such an augmenting subgraph, the algorithm then identifies components in the augmenting subgraph. Flow inside each component can be represented by a single variable. The flow balance constraints for all nodes that joint different components form a system of linear equations. Components joint with each other by nodes. Thus, the flow on an augmenting graph can be calculated by solving such a system of linear equations.

Wang and Yang [19] studied two special uncapacitated MDCPs by treating them as shortest path problems with additional constraints. They gave polynomial time solution methods and also discussed a network compaction scheme that can compact an uncapacitated distribution network to an equivalent one of smaller size. Wang and Lin [18] gave the first modified network simplex method for solving the MDCP in a graphical way. They described detailed graphical properties corresponding to a basis, and provided a solid theoretical foundation to the correctness of the multi-labeling method of Sheu et al. [16]. By iteratively solving a smaller system of equations, all the simplex pivot operations can be conducted graphically by their method. Similar work has also been proposed by Bahceci and Feyzioglu [3]. Note that most of these works assume simplified network structure in which the D-nodes are separated to each other. Although the proposed solution methods in the literature may still work for general distribution networks, their efficiency can be further boosted up by using our proposed network compaction scheme. Since the compaction scheme for uncapacitated distribution networks has been discussed by Wang and Yang [19], here in this paper we focus on the compaction scheme for capacitated distribution networks.

Most related research in the literature (e.g. [8], [16], [14], [19], and [14]) focus on the theoretical aspects of the solution methods without empirical analysis from computational experiments. To conduct extensive computational tests for evaluating the efficiency of proposed solution algorithms, a large number of distributed networks have to be generated and tested. However, conventional random network generators such as NETGEN [10], GENRMF [9], WASHINGTON [2] and AK [6] can only generate random networks composed by ordinary nodes. In order to produce random distribution networks for computational tests, in this paper we implement a random network generator which produces a compacted and acyclic distribution network.

The rest of this paper is organized as follows: Section 2 introduces definitions and notations; Detailed procedures of our capacitated network compaction scheme as well as its complexity are illustrated in Section 3; Section 4 explains procedures and

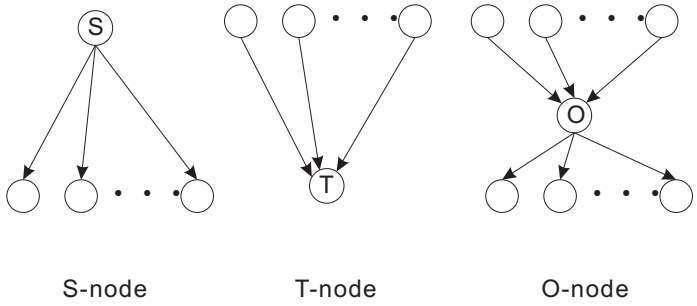

FIGURE 4. Illustration for an S-node, a T-node and an O-node in ordinary network

mathematical foundations to our proposed random distribution network generator; Finally, Section 5 summarizes and concludes the paper.

2. **Preliminaries.** Let $G = (N, A)$ be a directed graph with node set N and arc set A. We consider a capacitated network G where each arc (i, j) in A is associated with a nonnegative capacity u_{ij}. In ordinary network flow problems, there are three types of nodes: S-node, T-node and O-node (see Figure 4). 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. For each node i, we associate an integer $b(i)$ representing its supply/demand. In particular, $b(i) > 0$ if i is a supply node; $b(i) < 0$ if i is a demand node; and $b(i) = 0$ for each transshipment node i. Usually an S-node is a supply node, a T-node is a demand node, and any transshipment is an O-node.

A network flow model has to obey the constraints for flow balance and bounds. In particular, for each node i, the amount of flow enters i minus the amount of flow leaves i should equal to b_i. Also, the flow passing through an arc (i, j) can not exceed its capacity u_{ij}, but must be more than its lower bound l_{ij}. In this thesis we assume $l_{ij} = 0$ for each arc (i, j). The traditional maximum flow problem seeks the maximum amount of flow, denoted by v, that is shipped from the S-node to the T-nodes while the flow assignment satisfies the arc capacities and flow balance constraints for all arcs and nodes. Let N_S, N_T and N_O denote the set of S-nodes, T-nodes, and O-nodes, respectively. Besides these three types of ordinary nodes (i.e. S-nodes, T-nodes, and O-nodes), the distribution network model further introduces a type of distillation node, denoted by D-node, as the gray half-circle previously appeared in Figure 2. A D-node $i \in N_D$ has only one incoming arc (l, i) and at least two outgoing arcs called *member arcs*. Each member arc (i, j) is associated with a positive real number, denoted by k_{ij}, to specify the percentage of the flow in (l, i) that is to be distributed into the member arc (i, j). We also assume $\sum_{(i,j)\in A(i)} k_{ij} = 1$ $\forall i \in N_D$ and $k_{ij} > 0$ for each member arc (i, j). Figure 5 illustrates a D-node i by a half-circle. Let x_{ij} represents the flow in arc (i, j), then the flow along each member arc (i, j) for a D-node i with one incoming arc (l, i) can be calculated by $x_{ij} = k_{ij}x_{li}$. We call the relationship for flows on arcs connecting a D-node as "flow distillation", and k_{ij} as a *distillation factor*. For each member arc (i, j), we define a *normalized capacity* $\overline{u}_{ij} = u_{ij}/k_{ij}$ to represent the least amount of flow required in arc (l, i) to saturate arc (i, j).

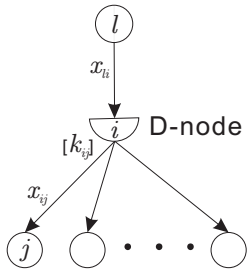

FIGURE 5. A D-node example

A *D-family* is a group of nodes other than D-nodes adjacent to the same D-node. We call the node in a D-family that sends flow to the D-node as its *mother node*, and the nodes that receive flow from the D-node in a D-family as its *member nodes*. Within a D-family, the *mother arc* connects the mother node to the D-node, while a *member arc* connects the D-node to a member node. By definition of a D-node, there is only one D-node, one mother node, and at least two member nodes in a D-family. For example, nodes 1, 2, 3, and 4 in Figure 6(a) form a D-family in which node 1 is the mother node, nodes 3 and 4 are member nodes, arc $(1, 2)$ is the mother arc, and arcs $(2, 3)$ and $(2, 4)$ are member arcs.

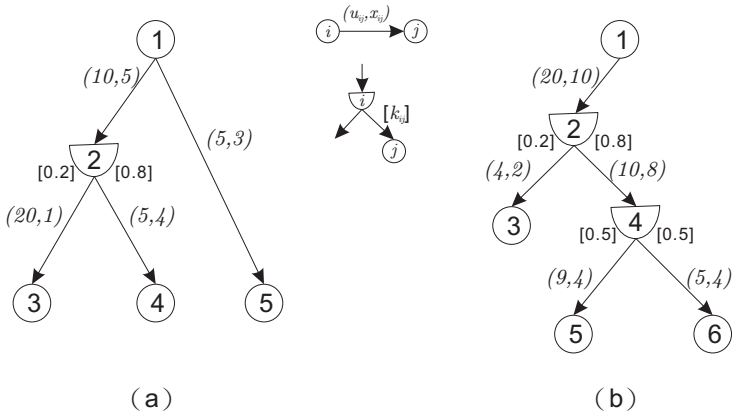

FIGURE 6. Two examples of distribution networks

For cases where a D-node is adjacent to anther D-node, by definition they can not form a D-family. We call the set containing the maximum number of adjacent D-nodes as a *D-group*. For example nodes 2 and 4 in Figure 6(b) form a D-group.

Here we use the formulation of the distributed maximum flow problem, as shown below, to explain how the distillation factor affects flows:

$$\max \sum_{i \in N_S} b_i \tag{1}$$

$$s.t. \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = b_i \begin{cases} \geq 0 & \forall\, i \in N_S \\ = 0 & \forall\, i \in N_O \text{ or } N_D \\ \leq 0 & \forall\, i \in N_T \end{cases} \tag{2}$$

$$0 \le x_{ij} \le u_{ij} \quad \forall (i,j) \in A \tag{3}$$

$$x_{ij} = k_{ij} x_{li} \quad \forall (i,j) \in A,\ (l,i) \in A \text{ and } i \in N_D \tag{4}$$

Unlike the traditional maximum flow problem that usually has optimal integral flows for cases of integral b_i and u_{ij}, the optimal solution for this linear programming problem is usually not integral, based on the proportional flow constraints 4. For cases containing adjacent D-nodes, the solution methods proposed in the literature (e.g. the multi-labeling method by Sheu et al. [16]) contain inefficient recursive steps that could be saved, should a D-group be treated as a equivalent single D-node. In next section, we will explain how our proposed network compaction scheme can help simplify the network structure and speed up solution method. Moreover, a simplified distribution network provides a better theoretical foundation for designing graphical algorithms for calculating optimal flows on distribution networks (see Wang and Lin [18] for details).

3. A network compaction scheme.

A distribution network may contain topology that is reducible. Here we give a preprocessing procedure to compact the original network to an equivalent one of smaller size, so that the solution methods on a compacted distribution network becomes more efficient.

3.1. Compacting rules.

We observe that a distribution network may be further simplified by either merging several nodes or arcs, or by unifying the effects of the capacities for some arcs. In particular, we give five rules to detect whether a distribution network is compactible, and then explain the compacting procedures.

(C1): Compacting D-groups.

Nodes in the same D-group can be compacted into a single D-node. A D-group contains the maximum set of adjacent D-nodes. Since flows over all arcs adjacent to a D-node are proportional to each other by some constants, we can easily calculate the flows entering an adjacent D-node which then can be used to calculate the flows on all of its adjacent arcs. Therefore the arcs adjacent to a D-group have flows proportional to each other. The process of calculating flows and resetting capacities on all arcs adjacent to a D-group have the same result as merging all adjacent D-nodes. Thus an entire D-group can be replaced by a single representative D-node. Suppose there are q_r O-nodes $(i_{q_1}, i_{q_2}, ..., i_{q_r})$ adjacent to a D-group with a representative D-node i_o. Suppose an O-node i_{q_w} is connected to i_o by a path $p_{i_o i_{q_w}}$ (i.e.$i_o \to ... \to i_{q_w}$) composed of $\left|p_{i_o i_{q_w}}\right|$ member arcs. The compacting process is as follows:

1. Among all the adjacent D-nodes, we retain the one (i_o) closest to the S-node and use i_o to represent the final merged D-node. For example, node 2 in Figure 7(a) is retained to be the representative D-node for that D-group.
2. Delete all original member arcs associated with this D-group and add new member arcs from $\{(i_o, i_{q_1}), (i_o, i_{q_2}), ..., (i_o, i_{q_r})\}$. In Figure 7(b), we create new arc $(2,5)$ and arc $(2,6)$.
3. The distillation factors on new member arcs can be calculated by multiplying the distillation factors of member arcs passing through intermediate D-nodes. In particular, $k_{i_o q_w} = \prod_{(i,j) \in p_{i_o q_w}} k_{ij}$. For example, the distillation factor for the new arc $(2,5)$ in Figure 7(b) is $0.8 * 0.5 = 0.4$.

4. The capacities for the new member arcs can be calculated by

$$\min_{(y,y')\ lies\ on\ p_{i_o w'},\ (w',q_w)\ lies\ on\ p_{i_o q_w}} \left\{ u_{w'q_w},\ \prod_{(i,j)\ lise\ on\ p_{y'q_w}} k_{ij} u_{yy'} \right\}$$

For example, the capacity of arc $(2,5)$ in Figure 7(b) equals to $min\ \{u_{45},\ k_{45}u_{24}\}$.

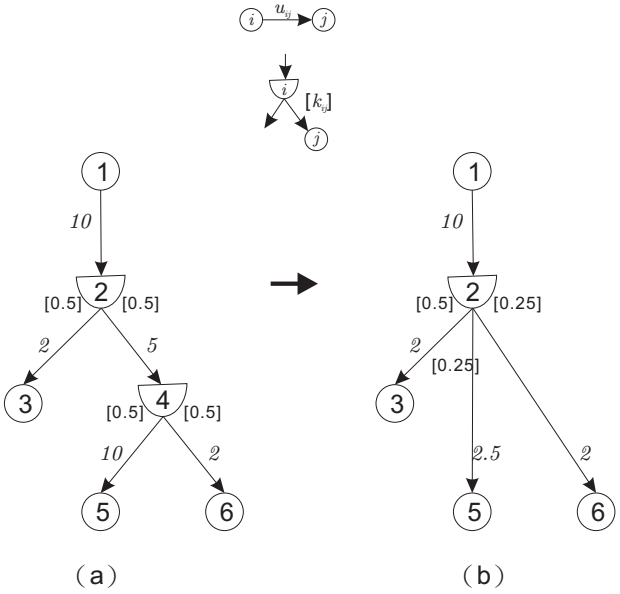

FIGURE 7. Adjacent D-nodes can be compacted into one D-node

Algebraically, flow distillation constraint $x_{45} = k_{45}x_{24}$ can be transformed into $x_{24} = \frac{x_{45}}{k_{45}}$, which can be plugged into the capacity constraint $x_{24} \leq u_{24}$. Thus we can combine the transformed constraint $x_{45} \leq k_{45}u_{24}$ with $x_{45} \leq u_{45}$ to set the new capacity for arc $(4,5)$ to be $min\{u_{45}, k_{45}u_{24}\}$. This compacting process thus removes some constraint.

It takes $O(m)$ time to apply this compacting rule by a search algorithm. Each time when we compact a D-group, at least one D-node and one arc connecting D-nodes will be reduced.

(C2): Compacting single transshipment O-nodes.

A single transshipment O-node is an O-node with only one incoming arc and one outgoing arc, and such a node can be compacted. When an O-node with single incoming and outgoing arcs, it is simply used for transshipment and has no affection to the flows in those arcs. Thus we may remove this O-node and merge these two arcs into one single arc with new capacity equal to the minimum capacity of the original two arcs. Take the distribution network in Figure 8(a) for example, after removing node 4 and merging arc $(2,4)$ and $(4,5)$ by an arc$(2,5)$ with capacity equal to $min\{8,4\} = 4$, the reduced graph is shown in Figure 8(b). Algebraically, this operation is to use a new flow variable x_{25} to replace the original flow variables x_{24} and x_{45}. That is,

$x_{25} = x_{24} = x_{45}$. The capacity constraints $x_{24} \leq u_{24}$ and $x_{45} \leq u_{45}$ can be integrated to set the new capacity for variable u_{25} to be $min\{u_{24}, u_{45}\}$. Thus we add one variable and one capacity constraint but remove two variables and two capacity constraints.

This compacting process takes $O(n)$ time since we only need to scan each node once. Each time when we compact a single transshipment O-node, this O-node and one arc connecting this O-node will be reduced.

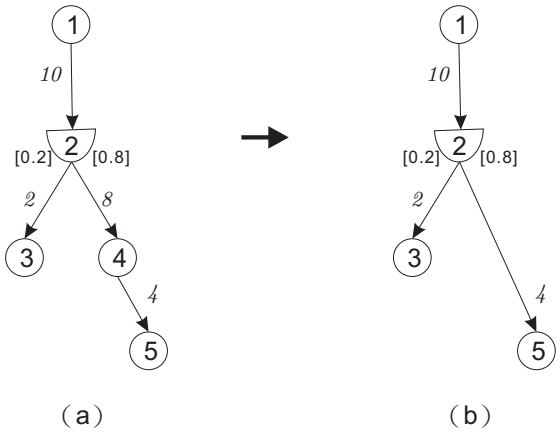

FIGURE 8. Any O-node with only one incoming arc and one outgoing arc can be compacted

(C3): Compacting self-loop arcs.

An arc with the same end nodes can be compacted. Although assuming no self-loop arcs in the network, they may be formed in some intermediate compacting process. Self-loop arcs are cycles in the network; therefore, they could be eliminated totally. See Figure 9 for example. Algebraically, the flow balance constraint for node 2 is $x_{12} + x_{22} = x_{23} + x_{22}$, which equals to $x_{12} = x_{23}$.after eliminating x_{22}. Thus this compacting process removes one variable and takes $O(m)$ time by scanning all outgoing arcs for each node. Once we compact one self-loop arc, one arc will be reduced.

(C4): Compacting parallel arcs.

Parallel arcs connecting to the same end nodes can be compacted. Like self-loop arcs, although we assume no parallel arcs in the network, they may be formed in some intermediate compacting process as well. Since we may view those arcs with the same tail and head nodes as one huge arc, we merge these

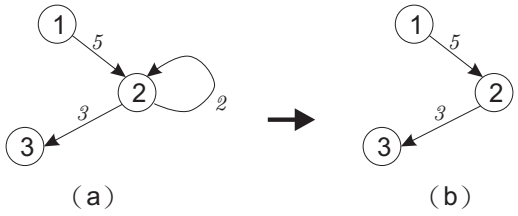

FIGURE 9. Self-loop arcs can be compacted

parallel arcs into one new arc. There are two cases for parallel arcs. One case is that the parallel arcs connect to the same end O-nodes and we sum up their capacities to be the capacity of new arc. See Figure 10 for example. Algebraically, this operation is to use a new flow variable x_{12} to replace the original flow variables x'_{12} and x''_{12} by $x_{12} = x'_{12}+x''_{12}$. The capacity constraints $x'_{12} \leq u'_{12}$ and $x''_{12} \leq u''_{12}$ can also be integrated to be a new capacity $u'_{12}+u''_{12}$ for variable u_{12}. The other case is that the parallel arcs connect to the same D-node and O-node. In this case, we merge parallel member arcs into a single member arc and calculate its distillation factor by summing up the distillation factors of these parallel member arcs. Then, the new capacity can be calculated by finding out the minimum normalized capacity in these parallel member arcs and multiplying it with the new distillation factor. See Figure 11 for examples. The distillation factor of new arc $(2,3)$ is $0.2+0.3=0.5$. The capacity of the new arc $(2,3)$ will be $min\left\{\frac{1}{0.2}, \frac{3}{0.3}\right\} * 0.5 = 2.5$. Algebraically, this operation is to use a new flow variable x_{23} to replace the original flow variables x'_{23} and x''_{23} as well. By specifying $k_{23} = k'_{23} + k''_{23}$, we may write $x_{23} = x'_{23}+x''_{23} = k'_{23}x_{12}+k''_{23}x_{12} = (k'_{23}+k''_{23})x_{12} = k_{23}x_{12}$. Also, the capacity constraint for the new arc can be integrated to be $x_{23} \leq min\left\{\frac{u'_{23}}{k'_{23}}, \frac{u''_{23}}{k''_{23}}\right\} * k_{23}$. In either case, one new variable and constraint is added to replace two variables and constraints.

This compacting process takes $O(m)$ time by scanning all outgoing arcs for each node. Each time when we compact parallel arcs, at least one arc will be reduced.

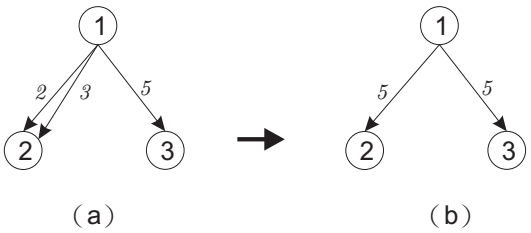

FIGURE 10. Parallel arcs connecting to the same end nodes can be compacted.

(C5): Compacting mismatched capacities.

Any D-node connected with arcs of mismatched capacities can be compacted. According to the flow distillation constraints, the flow on each outgoing arc from a D-node is proportional to the flow on the incoming arc. Therefore we may easily identify the first saturated arc for a D-node by calculating the normalized capacity for each outgoing arc relative to the incoming arc. Take Figure 12(a) as an example, x_{12} has to be at least $\frac{20}{0.2} = 100$ to saturate arc $(2,3)$, and to saturate arc $(2,4)$. Thus the bottleneck happens on arc $(2,4)$ since it is the first arc to be saturated when we increase x_{12}. By this observation we can identify the minimum normalized capacity $\overline{u}_{\min}$ among all arcs adjacent to a D-node i, reset the capacity of its incoming arc (l,i) to be $\overline{u}_{\min}$, and then reset the capacity for each member arc (i,j) by $k_{ij}\overline{u}_{\min}$. For the example in Figure 12(a), $\overline{u}_{\min} = \min\left\{15, \frac{20}{0.2}, \frac{8}{0.8}\right\} = 10$. Thus arc $(2,4)$ is the bottleneck and we can reset $u_{12} = \overline{u}_{\min} = 10$ and $u_{23} = 0.2\overline{u}_{\min} = 2$ as

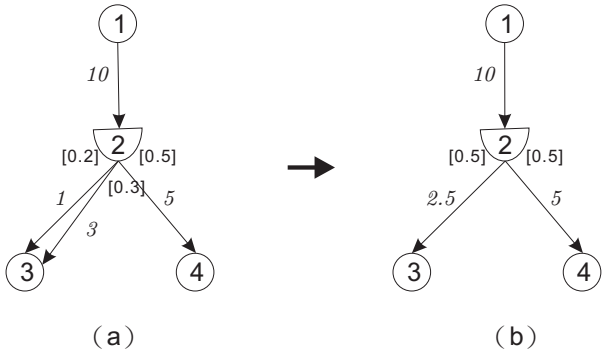

FIGURE 11. Parallel arcs connecting to the same D-node and O-node can be compacted.

shown in Figure 12(b). Algebraically, flow distillation constraints $x_{23} = k_{23}x_{12}$ and $x_{24} = k_{24}x_{12}$ can be plugged into the capacity constraints $x_{23} \leq u_{23}$ and $x_{24} \leq u_{24}$ to make $k_{23}x_{12} \leq u_{23}$ and $k_{24}x_{12} \leq u_{24}$. Using these two constraints, together with the original capacity constraint $x_{12} \leq u_{12}$, we can set the new capacity for arc $(1, 2)$ to be $min\left\{u_{12}, \frac{u_{23}}{k_{23}}, \frac{u_{24}}{k_{24}}\right\}$ and then reset the capacities for all the member arcs according to their distillation factors. This compacting process simplify the problem since once an arc adjacent to a D-node is saturated, all the other arcs will also be saturated. Thus a maximum flow algorithm may detect a bottleneck arc much earlier on this compacted network than the original one. This compacting process takes $O(m)$ time, but only needs to be conducted once in the end of the compacting process.

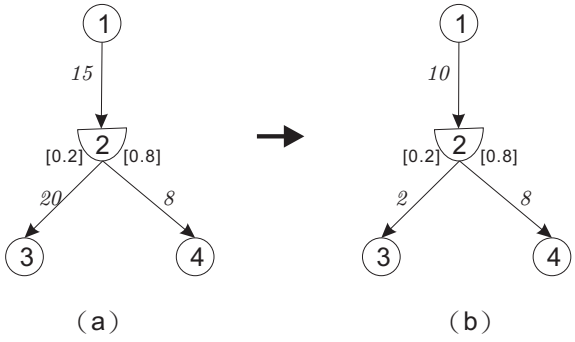

FIGURE 12. Any D-node with mismatch capacities of arcs can be compacted

3.2. **Inducing relations between compacting rules.** These five compacting rules may induce each other. We list all possibilities as below.

1. **Compacting D-groups may induce parallel arcs.**
   After compacting a D-group, the newly created member arcs may connect with the same O-node, which induces parallel arcs. See Figure 13 for example.
2. **Compacting a single transshipment O-nodes may induce a D-group.**
   When both of the arcs adjacent to a transshipment O-node that connect to

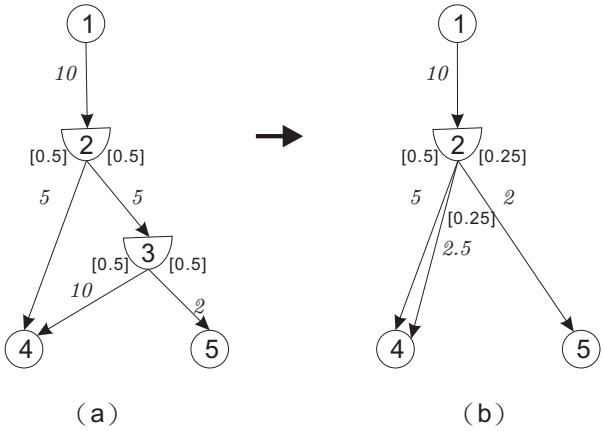

FIGURE 13. Compacting D-group may induce parallel arcs.

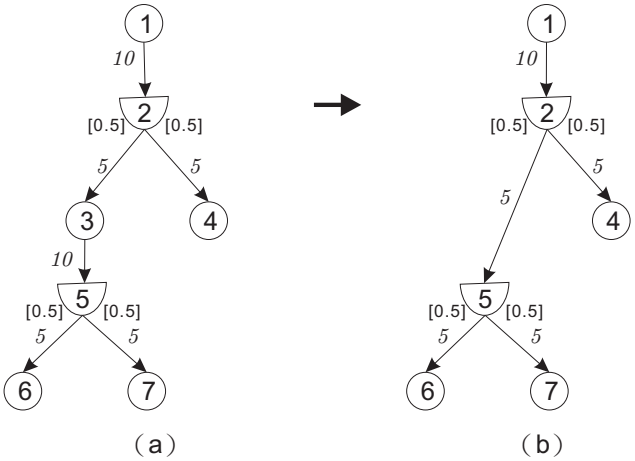

FIGURE 14. Compacting a single transshipment O-node may induce a D-group.

two D-nodes with its incoming arc and outgoing arc, compacting such an O-node will induce a D-group. See Figure 14 for example.

3. **Compacting a single transshipment O-node may induce mismatched arcs.**
   When one of the arcs adjacent to a single transshipment O-node that connects with a D-node, compacting such a single transshipment O-node may induce mismatched arcs. See Figure 15 for example.
4. **Compacting a single transshipment O-node may induce self-loop arcs.**
   After compacting a single transshipment O-node, an arc may connect to the same end nodes. See Figure 16 for example.
5. **Compacting a single transshipment O-node may induce parallel arcs.**
   After compacting a single transshipment O-node, some arcs may connect to the same end nodes. See Figure 17 for example.

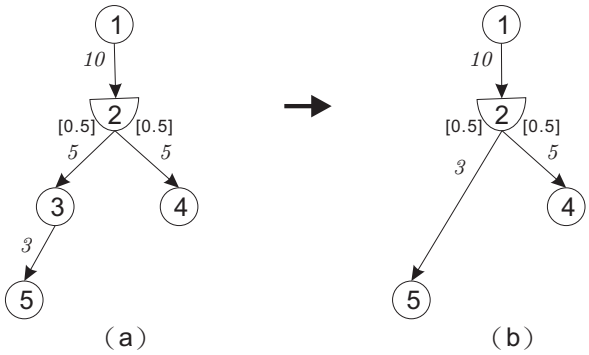

FIGURE 15. Compacting a single transshipment O-node may induce mismatched arcs.

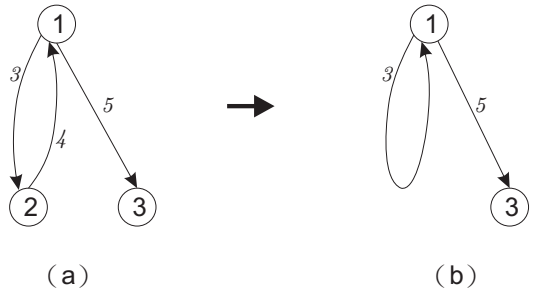

FIGURE 16. Compacting a single transshipment O-node may induce self-loop arcs.

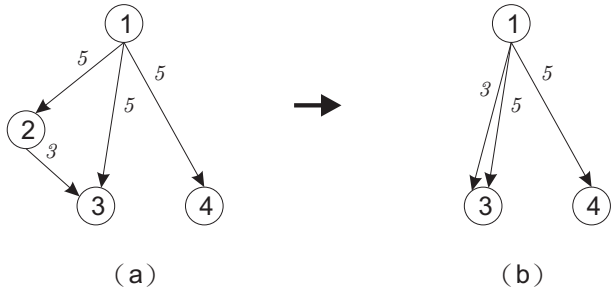

FIGURE 17. Compacting a single transshipment O-node may induce parallel arcs.

6. **Compacting parallel arcs may induce a single transshipment O-node.** After compacting parallel arcs, some nodes adjacent with more than two arcs may become a single transshipment O-node. See Figure 18 for example.
7. **Compacting self-loop arcs may induce a single transshipment O-node.** After compacting self-loop arcs, some nodes adjacent with more than two arcs may become a single transshipment O-node. See Figure 19 for example.

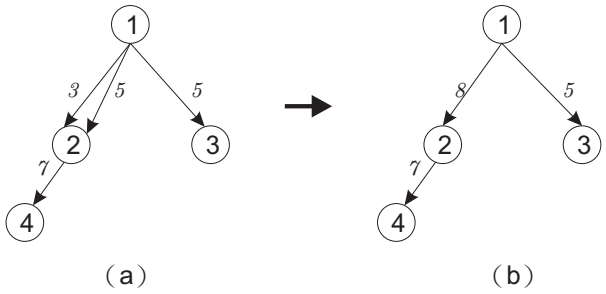

FIGURE 18. Compacting paralell arcs may induce single transshipment O-node.

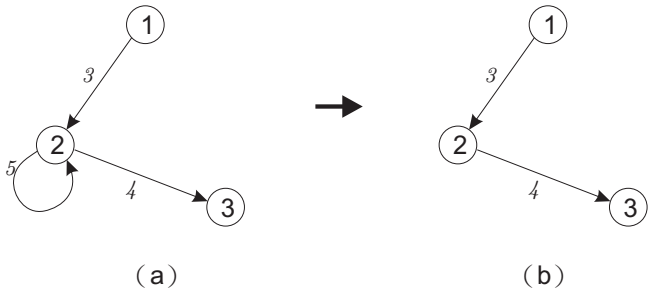

FIGURE 19. Compacting self-loop arcs may induce a single transshipment O-node.

Figure 20 illustrates the inducing relations between these five compacting rules. Note that there are three elementary directed cycles: (C1)-(C4)-(C2)-(C1), (C2)-(C3)-(C2) and (C2)-(C4)-(C2). Thus we may iteratively conduct (C1), (C4) and (C2), (C2) and (C3) or (C2) and (C4) until they finish their inducing relations and then conduct (C5) in the end of the process.

Each time when any of compacting rules (C1), (C2), (C3) or (C4) is applied, the network will be reduced by at least one arc or one node. Thus the entire compacting process takes $O(m^2)$ time and produces a simplified network which has smaller size and is easier to detect the bottleneck for solve a maximum flow problem.

In summary, we observe two properties for a compacted distribution network. First, every intermediate node (i.e. an O-node or a D-node) is adjacent to at least three arcs, including one incoming arc and one outgoing arc. Second, each D-node will not be adjacent to any other D-node. By these observations we may easily check whether a distribution network is compactible. For a distribution network not in its compact form, we may compact it using the following algorithm, which can not only reduce the size and simplify the structure of the original network but also shorten the time in calculating arc flows and speed up the solution procedures.

4. **A distribution network random generator.** This section explains how to generate random distribution networks for computational test. Popular network generators in the literature or on the internet such as GENRMF [9], WASHINGTON [2] and AK [6] that generate random max-flow problems only produce ordinary networks that contain only S-nodes,.T-nodes, and.O-nodes. Here we implement a

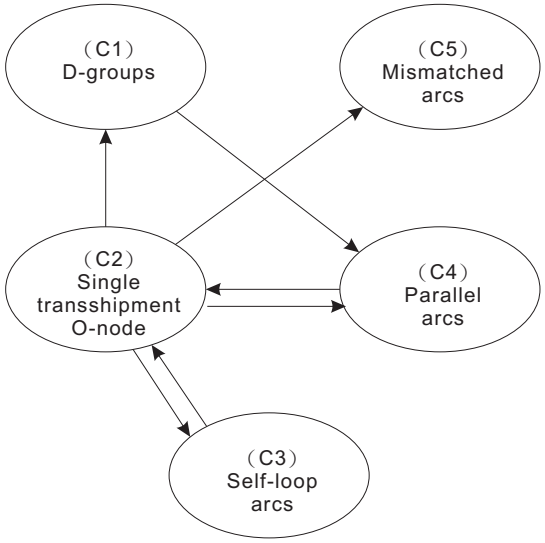

FIGURE 20. Inducing relations of different compacting rules

Algorithm 1 Compact(G)

while (G is not in compact form)
case 1: detect a D-group then apply (C1)
case 2: detect a single transhipment O-node then apply (C2)
case 3: detect self-loop arcs then apply (C3)
case 4: detect parallel arcs then apply (C4)
End while
case 5: detect capacity-mismatched arcs then apply (C5)

random network generator which produces a compacted and acyclic distribution network.

There are several parameters such as n (the total number of nodes), $u_{\max}$ (the maximum capacity for an arc) and $(outdeg\_D_{\max}, outdeg\_O_{\max})$ (the maximum outgoing-degree for a D-node and an O-node) that a user can specify to generate suitable networks. These parameters can not be arbitrarily set. For example, n must be larger than 3 because three nodes can not form a compacted network. Besides, $outdeg\_D_{\max}$ must be larger than 1 because each D-node has at least two outgoing arcs. The network should be connected, so every node except the T-node has at least one outgoing arc. For this reason, $outdeg\_O_{\max}$ must be larger than 0.

We group the nodes into two sets USE and $UNUSE$. The set USE contains the nodes that have been added into the network, while $UNUSE$ refers to the nodes that still wait to be added into the network. Every node i has a distance label $d(i)$ indicating its distance away from the source. We use the maximum number of arcs from the source to a node as its distance label. The notations used for constructing a random network generator are summarized as below.

$at\_least\_one\_D$: flag to record if there is any D-node
$d(i)$: the distance from source to node i.
USE: the set of nodes that has been put into the network

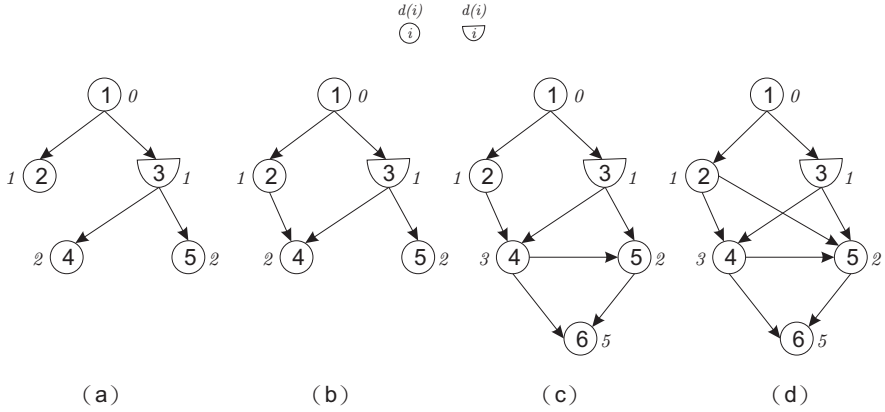

FIGURE 21. An example random network generated in some intermediate stage.

$UNUSE := N \setminus USE$: the set of nodes that have not been put into the network,
$LEAF$: the set of nodes in USE without outgoing arcs
$FARTHER(i)$: the set of O-nodes in USE with distance label equal to or larger than $d(i)$
$AVA(i) := UNUSE \cup FARTHER(i)$: the set of nodes that node i can connect to,
$outdeg(i)$: the outgoing-degree for node i
$outdeg_{\max}(i)$: the maximum outgoing-degree for node i
$outdeg_{\min}(i)$: the minimum outgoing-degree for node i

4.1. **Constructing a random distribution network.** We describe how to generate a random network as below.

1. **Remove a node i from $LEAF$ and construct $AVA(i)$.**
   Since our generator generates an acyclic network, an admissible node that one could connect to must be either an O-node of equal or larger distance labels in the current network or a node in $UNUSE$. In particular, a D-node in the current network is not admissible since it has only one incoming arc (and thus it can not receive flow from another node). $AVA(i)$ is used to denote the set of admissible nodes which node i could connect to. Suppose a random network generated in some intermediate stage is shown in Figure 21(a). The number besides a node i denotes its distance label $d(i)$. Suppose we set $outdeg\_O_{\max} = 4$ and $UNUSE = \{6\}$. The set of nodes in the current network that node 2 could connect to should be node 4 and node 5, denoted by $FARTHER(2) = \{4, 5\}$, since $d(4) \geq d(2)$ and $d(5) \geq d(2)$. Node 3 does not belong to $FARTHER(2)$ because it is a D-node. Therefore, $AVA(2) = \{4, 5, 6\}$.
2. **Decide the head nodes for arcs outgoing from node i.**
   Since n and $(outdeg\_D_{\max}, outdeg\_O_{\max})$ are already specified by the user and we have to make sure every node has sufficient admissible nodes to connect with, we set $outdeg_{\max}(i)$ to be $\min\{|AVA(i)|, outdeg\_D_{\max}\}$ if i is a D-node, or $\min\{|AVA(i)|, outdeg\_O_{\max}\}$ if i is an O-node. Because each D-node must have at least two outgoing arcs and each O-node must have at least one outgoing arc, we set $outdeg_{\min}(i)$ to be 2 if i is a D-node, or 1 if

i is an O-node. We may then generate a random integer $outdeg(i)$ from the range $[outdeg_{\min}(i), outdeg_{\max}(i)]$ to be its number of outgoing arcs, then we randomly pick head nodes from $AVA(i)$ for these outgoing arcs. See Figure 21(a) for example. $outdeg_{\max}(2) = min\{|\{4,5,6\}|, 4\} = 3$. Then we may generate $outdeg(2)$ from $[1,3]$. Suppose $outdeg(2) = 1$, and we select node 4 to be the head node of one outgoing arc from node 2. The result is illustrated in Figure 21(b).

3. **Update the distance label for a head node j that connects from node i.**
   Once a head node j is selected, $d(j)$ will be updated immediately. If node j is removed from $UNUSE$, we set $d(j) = d(i)+1$. Otherwise, node j must already exist in the current network with some finite distance label $d(j) \geq d(i)$. We then set $d(j) = max\{d(i)+1, d(j)\}$. Then we will check all reachable nodes from node j to see if their distance labels require updating. See Figure 21(b) for example. Since node 4 has an original distance label $d(4) = 2$, the updated distance label of node 4 will be $max\{d(2)+1, d(4)\} = max\{2,2\} = 2$. That is, $d(4)$ remains 2.
4. **Decide the node type for a head node j from $UNUSE$.**
   If a head node j is not from $UNUSE$, it must already exist in the current network and has a node type (i.e. either an O-node or a D-node). Otherwise, it is newly added to the current network and has to specify its node type. When $|AVA(j)|$ is 1, which means there is only one eligible head node for node j to connect to, so node j must be an O-node. When node j is the T-node, it must be an O-node. In other situations, the node type for node j will be randomly decided. In any case, a leaf should be an O-node and a D-node must connect with an O-node since the random network should be in a compact form. Therefore, if a new node which is just removed from $UNUSE$ is decided to be a D-node by the generator, we change it to be a non-leaf by immediately deciding all of its outgoing arcs and force their head nodes to be O-nodes. Otherwise, the new node just removed from $UNUSE$ is decided to be an O-node and we will add it into $LEAF$.
5. **Modify the topology of the random network whenever necessary.**
   Repeat step 1 to 4 until $LEAF$ becomes empty. Since the node type is decided randomly, the generator may generate a network containing only O-nodes. In that case, the network will be erased and the generator will be re-run until a random network that contains a D-node is produced. Because we want to generate a compacted random distribution network, every incompact situation requires correction. In particular, when we detect a single transshipment O-node, the generator will add artificial arcs from a node of smaller or equal distance label to a single transshipment O-node. See Figure 21(c) for example. Suppose it contains a random network generated in the stage of empty $LEAF$. Since node 2 is a single transshipment O-node, we will randomly add an arc, say, from node 2 to node 5 to ensure the resultant network is in a compact form.

The pseudo code of our random network generator is presented as follows.

4.2. **Mathematical properties for proposed random network generator.** Our random network generator will randomly generate a distribution network of n nodes which contains one source node (S-node) and one sink node (T-node). Our

Algorithm 2 Network_Generator

```
at_least_one_D:=FALSE
While (at_least_one_D=FALSE) do
    Initialization
    While (LEAF ≠NULL) do
        Pop node i from LEAF
        If (i ≠ t) then
            Construct AVA(i)
            Decide the outgoing-degree for node i
            Choose head nodes randomly for node i
            For each head node j
                Update information
                If (j is from UNUSE) then
                    Construct AVA(j)
                    Decide node type
                    If (j is O-node) then
                        Put node j into LEAF
                    Else then
                        at_least_one_D:=TRUE
                        Decide the outgoing-degree for node j
                        Choose head nodes ks randomly for node j
                        Update information, set nodes ks as O
                        Put nodes ks into LEAF
    End while
    Correct single transshipment O-nodes
End while
```

Procedure: Initialization

```
s:=1
t:=n
nodeTable (s).distance:= 0
nodeTable (s).degree := 0
nodeTable (s).type := 'O'
nodeTable (s).headnode = {NULL}
distancetable (0).node = {s}
Put s into LEAF
```

Procedure: Construct AVA(i)

```
Add those O-nodes in the current network whose distance ≥ node i into AVA
Add those nodes in UNUSE into AVA
```

randomly generated network will always be acyclic since a directed cycle means the objects (might be some material or product) will be processed over and over again along a directed cycle which seems not applicable for real applications. Another reason for generating acyclic random networks is to guarantee the feasibility of a distributed max-flow problem. In order to guarantee the feasibility, each node in

Procedure: Decide the outgoing-degree for node i

If (i is an O-node) **then**
  If ($|AVA(i)| \geq outdeg\_O_{\max}$) **then**
    Pick a number between 1 ~ $outdeg\_O_{\max}$
  Else then
    Pick a number between 1 ~ $|AVA(i)|$
Else then
  If ($|AVA(i)| \geq outdeg\_D_{\max}$) **then**
    Pick a number between 2 ~ $outdeg\_D_{\max}$
  Else then
    Pick a number between 2 ~ $|AVA(i)|$

Procedure: Update information

Update nodeTable
Update distanceTable
Update arcTable

Procedure: Decide node type

If ($|AVA(i)| \geq 2$ and $i \neq t$) **then** randomly choose type from $\{O, D\}$
Else then type must be O

Procedure: Correct single transshipment O**-nodes**

For each node i, except for s and t
  If (degree of node $i = 2$) **then**
    Randomly pick another O-node j which is not adjacent to node i
    Add an arc from the node of smaller distance label to the node of larger distance label
    Update information

a randomly generated network must connect to the T-node by at least one path. Since our generator always connects a node of smaller distance label to a node of larger distance label, the generated network thus has to be acyclic since there exists no directed cycle.

We also guarantee our randomly generated networks must be in compact forms. In particular, the generator always connects a D-node to an O-node, randomly adds arcs to any single transshipment O-node, avoids parallel arcs and self-loops, and assigns random capacities on arcs connecting to each D-node according to their distillation factors to avoid mismatched arcs.

First we list several important properties for any random distribution network generated by our random network generator.

Property 4.1. *For each arc* (i, j) *in the generated network,* $d(i) < d(j)$

Proof. There are four possible situations when an new arc (i, j) is created: (a) when node $i \neq t$ with a finite $d(i)$ is removed from $LEAF$ and node j is removed from (a1) $UNUSE$ and $j \neq t$, then $d(j) := d(i) + 1$; (a2) $UNUSE$ and $j = t$, then $d(j) := n - 1$; (a3) $AVA(i) \setminus UNUSE$, then $d(j) = max\{d(i) + 1, d(j)\}$; (b) when (i, j) with $d(i) < d(j)$ is added to prevent i (or j) from being a single transshipment O-node. Therefore $d(i) < d(j)$ remains to hold. □

Property 4.2. *Each generated network is acyclic.*

Proof. If there exists a directed cycle $i_1 - i_2 - \ldots - i_k - i_1$ of k arcs, by Property 4.1 we obtain a contradiction $d(i_1) < d(i_2) < \cdots < d(i_k) < d(i_1)$. □

Property 4.3. *The S-node has no incoming arc and the T-node has no outgoing arc. For each node other than the S-node and the T-node, it must have both incoming and outgoing arcs.*

Proof. The generator guarantees that there is only one S-node and one T-node. Furthermore, the S-node has no incoming arc and the T-node has no outgoing arc by the setting of the generator.

For each node i other than the S-node and the T-node, it must be either an O-node or a D-node and initially stored in $UNUSE$. If i is an O-node when the generator creates an arc (k, i), it will be moved to $LEAF$, which means i must have at least one incoming arc. Since the generator removes one node from $LEAF$ in each iteration and there are finitely many nodes, $LEAF$ will eventually become empty. Thus each node i in $LEAF$ will eventually be removed when an arc (i, j) is created, which means i must have at least one outgoing arc. If i is a D-node when the generator creates an arc (k, i), it will directly be removed from $UNUSE$ and then the generator will identify its outgoing arcs (i, j)s, which means i must have at least one incoming arc and more than one outgoing arcs. Thus for each node other than the S-node and the T-node, it must have both incoming and outgoing arcs. □

Property 4.4. *Each generated network is connected with n nodes.*

Proof. There exists no isolated node in each generated network by Property 4.3. Suppose there exist two (or more) components $COMP_1$ and $COMP_2$ in a generated network. By Property 4.2 we know both $COMP_1$ and $COMP_2$ are acyclic, and thus there must exist at least one node in $COMP_1$ and one node in $COMP_2$ that has no outgoing arcs (or incoming arcs), which contradicts Property 4.3. Thus each generated network is connected with n nodes. □

Property 4.5. *There always exists a path from the S-node to each node and a path from each node to the T-node.*

Proof. Starting from a node i in a generated network, we may conduct a DFS algorithm along an outgoing arc (i, j) which labels j and then keeps labeling a node k along some arc (j, k) if it exists. The DFS will keep labeling an unvisited node until the T-node is labeled since there exists no directed cycle by Property 4.2 and each intermediate node other than the T-node has at least one outgoing arc by Property 4.3. Thus each node can reach the T-node in a generated network.

Similarly, we may apply a reverse DFS algorithm starting from a node i which searches along an incoming arc (k, i) and labels k in a reverse fashion. Using similar arguments as described in previous paragraph, the reverse DFS algorithm will

eventually reach the S-node, which means the S-node can reach any other node in a generated network. □

Property 4.6. *Each O-node except the S-node and the T-node in a generated network must be connected by at least three arcs which include at least one incoming arc and one outgoing arc.*

Proof. For each O-node other than the S-node and the T-node, by Property 4.3 we know it has at least one incoming arc and one outgoing arc. The generator will also check the existence of a single transshipment O-node (i.e. an O-node connected with exactly one incoming arc and one outgoing arc) and add new arcs for such an O-node so that in the end each O-node will have at least three adjacent arcs. □

Property 4.7. *There exist no self-loops, parallel arcs, arcs of mismatched capacities and D-groups in a generated network.*

Proof. Property 4.1 guarantees no self-loop.

When the generator decide an outgoing arc (i, j) for either an O-node i removed from $LEAF$ or a D-node i removed from $UNUSE$, it will always choose different head node j. When adding a new arc (k, i) or (i, j) to a single transshipment O-node i where node k or node j are O-nodes, the generator also prevent creating parallel arcs. Thus there exists no parallel arc in a generated network.

Whenever a D-node i is created from a mother node k which is an O-node, the generator will immediately connect i to at least two member nodes which are also O-nodes chosen from $AVA(i)$. So the generator will not form a D-group at any time.

When the generator assigns the capacity for arcs connecting to a D-node, it will be proportional to the assigned distillation factor and thus no arcs of mismatched capacity will be created. □

Property 4.8. *Each generated network is in a compact form.*

Proof. It follows directly from Property 4.6 and Property 4.7. □

We also list several properties that give more insights about our generator. We skip some trivial proofs.

Property 4.9. *Each node in $LEAF$ must be an O-node associated with at least one incoming but no outgoing arc.*

Property 4.10. *Throughout the generating process, each node with a finite distance label must already exist in the current network, whether it is in $LEAF$ or not. Furthermore, such a node is in $LEAF$, if and only if it has no outgoing arc at that time.*

Property 4.11. *In any intermediate stage, the node with the largest finite distance label must be in $LEAF$.*

Proof. Suppose node i is the node in the current network with the largest finite distance label and it is not in $LEAF$, then i must have an outgoing arc (i, j) connecting to a node j with finite distance label $d(j) > d(i)$ by Property 4.1. This contradicts the assumption that $d(i) \geq d(j)$. □

Property 4.12. *$UNUSE$ will become empty before $LEAF$ becomes empty.*

Proof. Suppose $LEAF$ contains only one node i which is not the T-node (otherwise it is trivial). Node i must be an O-node by Property 4.9. When we remove i from $LEAF$, we will immediately identify an outgoing arc (i, j) leaving from i, and j must be removed from $UNUSE$ instead of $FARTHER(i)$ (which means j is in $LEAF$) since otherwise $d(j) \geq d(i)$ by Property 4.11 which contradicts the assumption that i is the only node in $LEAF$. □

Property 4.13. *Each O-node stays in $LEAF$ once. A D-node j connected from an O-node i must be removed directly from $UNUSE$, instead of $FARTHER(i)$.*

Property 4.14. *Both $LEAF$ and $UNUSE$ will eventually become empty.*

Proof. The generator removes one node from $LEAF$ in every iteration. No node will be added to $LEAF$ and $UNUSE$ throughout the generating process. Since there are finitely many nodes, each node stays at $LEAF$ or $UNUSE$ at most once and the generator removes at least one node from either $UNUSE$ or $LEAF$ at each iteration, by Property 4.12 finally $LEAF$ and $UNUSE$ will become empty. □

Property 4.15. *The T-node is the last node removed from $UNUSE$, but may not be the last node removed from $LEAF$.*

Property 4.16. *Whenever a node i is to be removed from $LEAF$, it must be able to connect to some node j with $d(j) \geq d(i)$.*

Property 4.17. *For each generated D-node, there must exist sufficient number of O-nodes to be its member nodes.*

Proof. For an unknown-type node i, only when $|AVA(i)| \geq 2$, it will become eligible to be a D-node. Once node i is decided to be a D-node, its upper bound of outgoing degrees, $outdeg_{\max}(i)$, will be $\min\{|AVA(i)|, outdeg\_D_{\max}\}$, which guarantees at least two outgoing arcs for node i. □

5. **Conclusions.** Optimization problems in a distributed network are important and applicable for real applications. For example, a manager of a logistics company may estimate the maximum amount of materials purchasable or products manufacturable for the supply chain network of his company by solving a distributed max-flow problem.

This paper focus on issues in solving the distributed max-flow problems. We first contribute a polynomial time network compaction algorithm in section 3 which can be used as a preprocessing procedures to transform the original problem into an equivalent problem of smaller size. The techniques of our network compaction algorithm can also be generalized to solve the MDCP which are min-cost network flow problems on distribution networks. Another major contribution of this paper is the construction of a random distribution network generator in section 4. With our random network generator, we are able to generate compacted distributed max-flow problems for computational tests and evaluation. Again, the techniques we developed for our random network generator can also be generalized for producing compacted distribution networks for min-cost distribution network flow problems. The theoretical properties of our network generator that we have demonstrated and shown in section 4 are useful and can be used for related future research.

For future research, we suggest investigating real-world applications that exploiting the proportional flow constraints. For example, to model the traffic flows of bike sharing systems, Shu et al. [17] proposed a specialized network flow model in which

the bikes departing from an origin rental site are distributed to different destination sites along outgoing arcs in the time-space network. In this case, almost all intermediate nodes in the time-space networks are D-nodes, and our network compaction scheme will be very helpful. We also suggest to investigate the reliability problems on distribution networks (e.g. [13]). We expect our compaction scheme should help boost up the algorithmic efficiency of calculating the network reliability.

Acknowledgments. I-Lin Wang was partially supported by the National Science Council of Taiwan under Grant NSC102-2221-E-006-141-MY3.

REFERENCES

[1] R. K. Ahuja, T. Magnanti and J. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] R. J. Anderson and J. C. Setubal, Goldberg's algorithm for maximum flow in perspective: A computatioinal study, in *Network flows and matching: First DIMACS implementation challenge* (eds. D. S. Johnson and C. McGeoch), 12, American Mathematical Society, (1993), 1–17.

[3] U. Bahceci and O. Feyzioglu, A network simplex based algorithm for the minimum cost proportional flow problem with disconnected subnetworks, *Optimization Letters*, **6** (2012), 1173–1184.

[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] B. V. Cherkassky and A. V. Goldberg, On implementing push-relabel method for the maximum flow problem, *Algorithmica*, **19** (1997), 390–410.

[7] B. T. Denton, J. Forrest and R. J. Milne, Ibm solves a mixed-integer program to optimize its semiconductor supplychain, *Interfaces*, **36** (2006), 386–399.

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

[9] D. Goldfarb and M. D. Grigoriadis, A computational comparison of the dinic and network simplex methods formaximum flow, *Annals of Operations Research*, **13** (1988), 83–123.

[10] D. Klingman, A. Napier and J. Stutz, Netgen: A program for generating large scale capacitated assignment, transportation and minimum cost flow networks, *Management Science*, **20** (1974), 814–820.

[11] J. Koene, *Minimal Cost Flow in Processing Networks, a Primal Approach*, PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 1983.

[12] L.-C. Kung and C.-C. Chern, Heuristic factory planning algorithm for advanced planning and scheduling, *Computers and Operations Research*, **36** (2009), 2513–2530.

[13] Y.-K. Lin, C.-T. Yeh and C.-F. Huang, Reliability evaluation of a stochastic-flow distribution network with delivery spoilage, *Computers and Industrial Engineering*, **66** (2013), 352–359.

[14] H. Lu, E. Yao and L. Qi, Some further results on minimum distribution cost flow problems, *Journal of Combinatorial Optimization*, **11** (2006), 351–371.

[15] P. Lyon, R. J. Milne, R. Orzell and R. Rice, Matching assets with demand in supply-chain management at ibm microelectronics, *Interfaces*, **31** (2001), 108–124.

[16] 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.

[17] J. Shu, M. Chou, Q. Liu, C.-P. Teo and I.-L. Wang, Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems, *Operations*, **61** (2013), 1346–1359.

[18] I. L. Wang and S. J. Lin, A network simplex algorithm for solving the minimum distribution cost problem, *Journal of Industrial and Management Optimization*, **5** (2009), 929–950.

[19] 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** (2009), 53–63.

Received March 2014; 1st revision May 2014; final revision November 2014.

E-mail address: ilinwang@mail.ncku.edu.tw