International Journal of Reliability and Quality Performance, 1(1) January-June 2009, pp. 53-63

On Solving the Uncapacitated Minimum Cost Flow Problems in a Distribution Network

I-Lin Wang* & Yu-Hui Yang

ABSTRACT

In this paper, we consider special minimum cost flow problems in a kind of manufacturing network recently introduced by Fang and Qi [5] called as a distribution network. A new kind of nodes, called *D*-nodes, are incorporated to describe a distilling operation that decomposes one raw-material to several products with fixed ratios. The arc capacity in our models is not limited so that the uncapacitated minimum distribution cost flow problems can be regarded as specialized shortest path problems. We define two special uncapacitated minimum cost flow problems: UMDCP_1 and UMDCP_2, give their formulations for the cases that satisfy the demand of one sink node from one source node, and then develop efficient solution methods based on Dijkstra's algorithm. A polynomial-time preprocessing procedure is also proposed to reduce the original problem into an equivalent one of smaller size and simplifies the solution procedures.

Keywords: Manufacturing network; Distribution network; Minimum cost flow problem; Shortest path

1. INTRODUCTION

Network flow problems appear everywhere in our daily life. They are used in practice to model many industry, transportation, and telecommunication problems. However, an ordinary network flow model has its limitation for modeling more complicated manufacturing scenarios, such as the synthesis of different raw-materials to one product or the distilling of one material to many different products. Fang and Qi [5] present a generalized network model called the *manufacturing network flow* (MNF) for this purpose. They also present a minimum distribution cost problem which involves uncapacitated arcs and four types of nodes: *O*-nodes, *S*-nodes, *T*-nodes and *D*-nodes to illustrate a special manufacturing process which focuses on distillation relationships between specific nodes. It differs from a traditional network model because a new kind of nodes, called the *D*-nodes, are incorporated to describe a distilling operation that decomposes one raw-material to several products with pre-specified ratios. In other words, besides the flow balance constraints associated with each node, the flows going out from a *D*-node have to obey the specified ratios of the flow entering a *D*-node. Fang and Qi [5] propose a modified network simplex algorithm to solve the uncapacitated minimum distribution cost problem (UMDCP).

This paper investigates two special UMDCPs based on the minimum distribution cost problem by Fang and Qi [5]. Our first problem (UMDCP_1) only considers the fixed costs in all processes in a manufacturing network. That is, when we calculate the cost on a network problem, we only consider whether an arc is used or not, rather than the flow value on an arc. In other words, no matter how many goods are produced or transshipped, we only consider the fixed cost of an operated manufacturing process (corresponding to an arc). If one material enters a special working station (i.e. a *D*-node), it has to produce several different products according to some predefined fixed ratios. Among those products made by a *D*-node, the system will only select one for further manufacturing processes but cease the processes for other products. Such situations may happen in practice. For example, this may be resulted from business contract which only allows one out of many products produced by a *D*-node to be further processed. The goal is to minimize the total fixed cost required to produce the set of the requested products. Instead of considering the fixed cost, our second problem (UMDCP_2) considers the unit cost and the flows in a manufacturing network. Like UMDCP_1, once a material enters a *D*-node, among those products made by a *D*-node, UMDCP_2 only allows

* Department of Industrial and Information Management, National Cheng Kung University, Tainan 701, Taiwan, *Email: ilinwang@mail.ncku.edu.tw*

one product to receive further processing, and seeks the minimum total unit costs to produce the set of requested products as well.

The structure of this paper is organized as follows: Section 2 introduces definitions and notations for our UMDCPs, together with literature review. Section 3 and Section 4 illustrate the one-to-one $UMDCP_1$ and one-to-one $UMDCP_2$, respectively. Section 5 gives a preprocessing operation to reduce the original problem into an equivalent one of smaller size. Section 6 concludes the paper.

2. PRELIMINARIES

Let $G = (N, A)$ be a general network where N and A are the node set and arc set, respectively. The numbers of nodes and arcs are $|N| = n$ and $|A| = m$. For each arc $(i, j) \in A$, c_{ij}, is its unit cost and x_{ij} is its flow. In the UMDCPs, all arcs are uncapacitated which means $0 \le x_{ij} \le \infty$. Note that we allow multiple materials (products) to flow through the network in our model, but only one material (product) on an individual arc.

For each node $i \in N$, we define the set of entering nodes to node i as $E(i) := \{ j \in N : (j, i) \in A\}$ and the set of leaving nodes from node i as $L(i) := \{j \in N : (i, j) \in A\}$. Let N_O, N_S, N_T, and N_D denote the set of O-nodes, S-nodes, T-nodes and D-nodes, respectively. Thus $N = N_o \cup N_S \cup N_T \cup N_D$. An O-node is an ordinary node for transshipment which may have several incoming arcs and outgoing arcs, but only one kind of material/semi-product can flow through it with balance. An S node is a source node for raw materials. A T-node is a sink node for final products. There may exist one or several source (or sink) nodes in a manufacturing network, and each of them represents one different raw material (or final product). A D-node is a distillation node which has only one incoming arc (for one kind of material) but multiple outgoing arcs (for different kinds of products), and the flow value on an outgoing arc is proportional to the flow value on the incoming arc. In particular, for each $i \in N_D$, we have $E(i) = \{i^*\}$ and $x_j = k_{ij} x_{i^*i}$ for each $j \in L(i)$ where k_{ij} is a pre-specified positive real number. We call this distillation relationship as a *flow distillation* constraint.

Fang and Qi [5] first discuss the following UMDCP:

$$\min \sum\nolimits_{(i,j)\in A} c_{ij} x_{ij} \tag{UMDCP}$$

$$s.t. \quad \sum\nolimits_{j\in E(i)} x_{ji} - \sum\nolimits_{j\in L(i)} x_{ij} = 0 \quad \forall i \in N_O \tag{1}$$

$$\sum\nolimits_{j\in L(i)} x_{ij} \le u_i \quad \forall i \in N_S \tag{2}$$

$$\sum\nolimits_{j\in E(i)} x_{ji} \ge d_i \quad \forall i \in N_T \tag{3}$$

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

where $\Sigma_{(i,j)\in A} k_{ij} = 1$ for each $i \in N_D$. They propose a network simplex algorithm to solve this problem. However, they do not give procedures for obtaining an initial basic feasible solution to start their method. Neither do they specify steps to update dual variables. Techniques in solving the conventional minimum cost network flow problems such as the network simplex algorithm can not be directly applied here due to the appearance of the distillation constraint (4). The distillation constraint also destroys the property of total unimodularity which makes the optimal flow not necessarily be integral.

In this paper, we focus on two problems similar to the UMDCP of Fang and Qi [5]. We first formulate our problems and then develop the solution methods based on Dijkstra's algorithm in next 2 sections.

3. MATHEMATICAL MODELS AND SOLUTION METHODS FOR $UMDCP_1$

Our first uncapacitated minimum distribution cost problem, $UMDCP_1$, finds the minimum cost to connect a source node (S-node) to one sink nodes (T-node) using arcs in a distribution network. In this problem, we only consider the dependency of arcs connecting to a D-node and ignore its flow distillations. Here we consider the one-to-one case that satisfies the demand of a specific sink node from a specific source node.

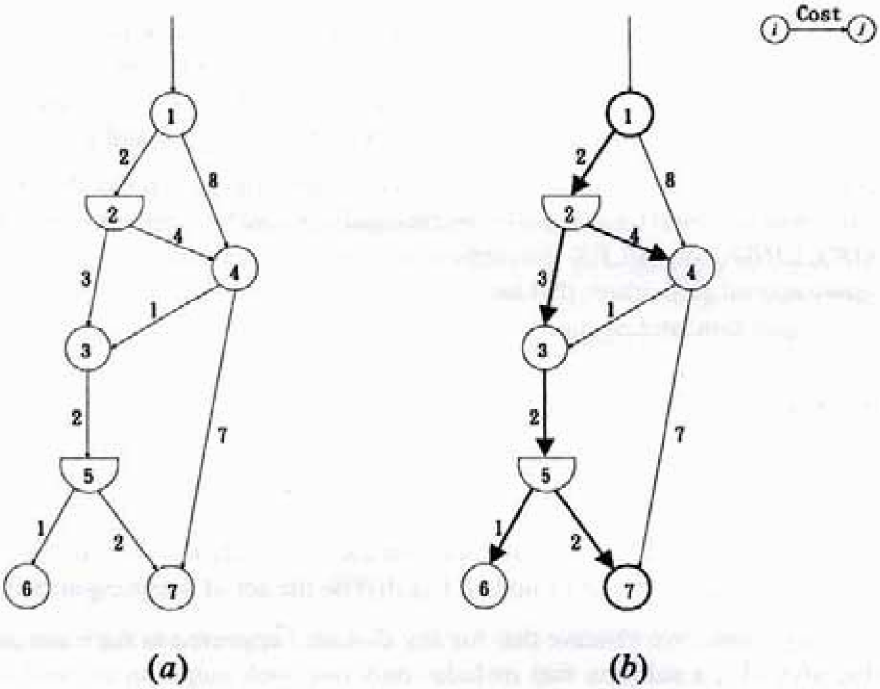


Figure 1: A One-to-one UMDCP$_1$ Example

For the one-to-one UMDCP$_1$, once a D-node is selected in a main path connecting the source s and sink t, all of its outgoing arcs also have to be selected. However, among these outgoing arcs, only one will be in the main path connecting s and t. A one-to-one UMDCP$_1$ example connecting node 1 and 7 is shown in Figure 1(a). Its optimal solution which contains arcs (1, 2), (2, 3), (2, 4), (3, 5), (5, 6) and (5, 7) is illustrated in Figure 1(b).

To formulate the one-to-one UMDCP$_1$, we associate each outgoing arc (i, j) of each D-node i with another parallel artificial arc $(i, j)'$ and set its length to be the same as c_{ij}. This parallel arc $(i, j)'$ is a copy of (i, j). When appears $(i, j)'$ in a solution to a one-to-one UMDCP$_1$, it means the D-node i is in the main path connecting s and t but the O-node j is not (e.g. j can be viewed as a side-product). On the other hand, if (i, j) appears in a solution, it means both the D-node i and O-node j are in the main path connecting s and t (e.g. j can be viewed as a major product). Thus arc (i, j) and arc $(i, j)'$ can not appear in a solution at the same time. If either (i, j) or $(i, j)'$ appears in a solution, it means the D-node i must appear in the main path and thus the fixed cost of (i, j) (or $(i, j)'$) has to be taken into consideration. For each sink node i in N_T, we set its demand $d(i) = 1$ when i is the requested sink node, otherwise $d(i) = 0$. Let B be the set of all arcs $(i, j)'$ from each $i \in N_D$ and $j \in L(i)$. We can formulate a UMDCP$_1$ problem as follows:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{(i,j)\in B} c_{ij}x'_{ij} \qquad \text{(UMDCP}_1\text{)}$$

$$s.t. \sum_{k\in L(i)} x_{ik} - \sum_{j\in E(i)} x_{ji} = 0 \quad \forall i \in N_O \qquad (5)$$

$$\sum_{j\in E(i)} x_{ji} = d(i) \quad \forall i \in N_T \qquad (6)$$

$$x_{ji} - x_{ik} - x'_{ik} = 0 \quad \forall j \in E(i), \forall k \in L(i), \forall i \in N_D \qquad (7)$$

$$\sum_{(i,k)\in A(i)} x_{ik} \le 1 \ \forall i \in N_D \qquad (8)$$

$$x_{ij}, x'_{ij} \in (0, 1) \ \forall (i, j) \in A \text{ and } \forall (i, j) \in B \qquad (9)$$

The objective function minimizes the total cost of all the original and artificial arcs in the network. The first constraint is the flow balance constraint for each O-node. The second constraint requires all T-nodes to satisfy their demands. The third and forth constraints are relative to D-nodes which show that either an original arc or its associated artificial parallel arc can be selected, and if an original arc from a D-node is selected, the parallel arcs of other outgoing arcs from the same D-node will also be selected. The fifth one means that all x_{ij} and x'_{ij} are binary. That is, arc (i, j) is selected, if $x_{ij} = 1$ or $x'_{ij} = 1$; arc (i, j) is not selected if $x_{ij} = 0$ and $x'_{ij} = 0$.

A general integer programming problem, including the 0-1 integer programming model of UMDCP$_1$, is potentially very difficult (that is, not in polynomial time) to solve, and is usually solved by commercial mathematical programming solvers such as LINDO, LINGO or CPLEX. Nevertheless, after more careful analysis, here we are able to solve UMDCP$_1$ by two combinatorial algorithms that run in polynomial time. In particular, Section 3.1 gives details of our modified Dijkstra's algorithm, and Section 3.2 converts a one-to-one UMDCP$_1$ to a shortest path problem.

3.1 A Modified Dijkstra's Algorithm for the One-to-one UMDCP$_1$

Let s be the only supply node in a distribution network $G = (N, A)$. S is the set of permanently labeled nodes. $\bar{S}$ is the set of temporarily labeled nodes, and $S \cup \bar{S} = N$. The distance label to any permanently labeled node represents the shortest distance from the source to that node. For any temporarily labeled node, the distance label is an upper bound on the shortest path from the source to that node. For each $i \in N$, denote $d(i)$ to be the distance from node s to node i, and pred (i) to be the predecessor of node i. Let $A(i)$ be the set of outgoing arcs from node i.

Before running our algorithm, we observe that for any O-node i appeared in the main path connecting s and t, among those outgoing arcs of i, a solution that includes only one such outgoing arc will have lower cost than a solution that includes more than one outgoing arcs of i. This observation helps to develop a more efficient algorithm similar to the conventional Dijkstra's algorithm. In particular, whenever we select an O-node to be in the main path, only one of its outgoing arcs can also be in the main path. The algorithm illustrated in Figure 2 selects a node closest to the source at each iteration. Using the predecessor information, the main path connecting s and t, together with all outgoing arcs of the D-nodes in the main path can be identified.

The correctness and complexity for this algorithm is exactly the same as the original Dijkstra's algorithm. A naive implementation of Dijkstra's algorithm takes $O(n^2)$ time. Dial's bucket implementation [3] gives better running time $O(m + nC)$ for graphs with small arc lengths where $C = \max_{(i,j)\in A} \{|C_{ij}|\}$. Ahuja *et al.* [2] use a radix heap data structure that shortens the time to check nonempty buckets and give a running time of $O(m + n \log nC)$. The binary-heap implementation by Johnson [7] takes $O(m \log n)$ time while the Fibonacci heap implementation [6] has the

Modified Dijkstra's Algorithm for one-to-one UMDCP

```
begin
    S: {s}; S̄: = N – S;
    d(s):= 0; d(i): = ∞ ∀i ∈ S̄;
    pred(i):=0 ∀i ∈ N;
    while S ≠ N do
        select the node i = arg min_j{d(j): j ∈ S̄};
        S:= S ∪ {i}; S̄: = S̄ – {i};
        if i ∈ N_O then
            for each arc (i, j) ∈ A(i) do
                if j ∈ N_D then
                    d(j):= d(i) + c_ij + Σ_(j,k)∈A(j) c_jk;
                    pred(j): = i;
                    for each arc (j, k) ∈ A(j) do
                        if d(k) > d(j) then
                        d(k) := d(j); pred(k) := j;
                else
                    if d(j) > d(i) + c_ij then
                        d(j):= d(i) + c_ij; pred(j): = i;
end
```

Figure 2: A Modified Dijkstra's Algorithm for Solving a One-to-one UMDCP$_1$.

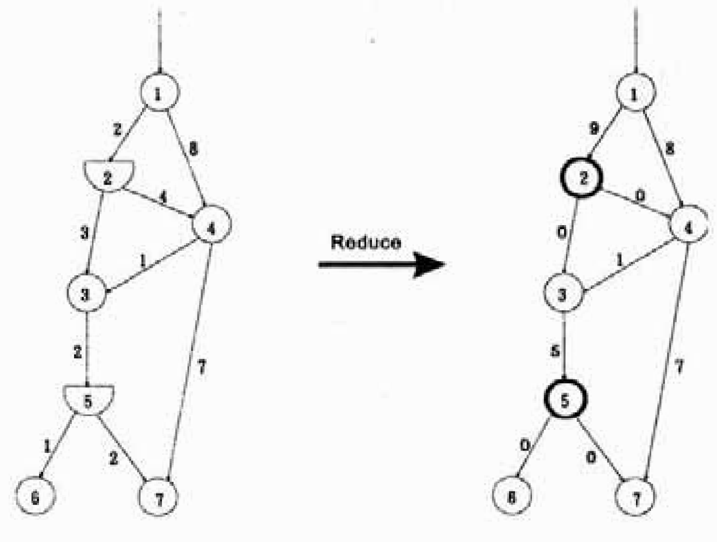
best $O(m + n \log n)$ running time. In general, the running time for a one-to-one shortest path algorithm is $S(n, m, C)$, a function of n, m and C [1].

Now we give a transformation procedure which converts a one-to-one UMDCP$_1$ into a shortest path problem directly solvable by the original Dijkstra's algorithm.

3.2 Converting a One-to-one UMDCP$_1$ to a Shortest Path Problem

We observe that the effect of a D-node in fact can be transferred to its incoming arc. In particular, when a D-node is selected in a main path from s to t, we also have to pay for the outgoing arcs of the D-node. Therefore, we may add all the arc lengths to the incoming arc of the D-node, and reset the lengths of all the outgoing arcs from the D-node to be 0. Then, all D-nodes can be transformed to be O-nodes, so we only need to consider the flow balance constraints for the transformed network which can be solved by Dijkstra's algorithm. Figure 3(b) shows a transformed distribution network from its original one in Figure 3(a).

Now we explain why the transformation is valid. Suppose a D-node has α outgoing arcs. By equation (8), we have $x_{ik_1} + x_{ik_2} + \ldots + x_{ik_\alpha} \leq 1$. If arc (i, k_1) is selected (i.e., $x_{ik_1} = 1$ and $x_{ik_1} = 0$), all other $x_{ik_p} = 0$ and $x'_{ik_p} = 1$ for $p = 2, \ldots, \alpha$. Thus, we know that $x_{ik_1} = x'_{ik_2} = \ldots x'_{ik_\alpha} = 1$ and $x'_{ik_1} = x_{ik_2} = \ldots = x_{ik_\alpha} = 0$. By equation (7), $x_{ji} = x_{ik} + x'_{ik}$ for all k, we obtain $x_{ji} = x_{ik_1} + x'_{ik_1}$, $x_{ji} = x_{ik_2} + x'_{ik_2}, \ldots, x_{ji} = x_{ik_\alpha} + x'_{ik_\alpha}$. Sum up these α equations, we have $\alpha x_{ji} = x_{ik_1} + x_{ik_2}$



(a) (b)

Figure 3: An Example to Convert a UMDCP$_1$ to a Shortest Path Problem

$+ \ldots + x_{ik_\alpha} + x'_{ik_1} + x'_{ik_2} + \ldots + x'_{ik_\alpha}$. Since $x_{ik_1} = x'_{ik_2} = \ldots = x'_{ik_\alpha}$ and $x'_{ik_1} = x_{ik_2} = \ldots = x_{ik_\alpha} = 0$, we have $\alpha x_{ji} = \alpha x_{ik_1}$. Hence, $x_{ji} = x_{ik_1}$ and $x_{ji} = x_{ik_1} + x_{ik_2} + \ldots + x_{ik_\alpha} = \sum_{(i,k) \in A(i)} x_{ik}$. This satisfies the flow balance constraint for an O-node.

Thus we can transform a one-to-one UMDCP$_1$ to be a traditional one-to-one shortest path problem and solve it by the shortest path algorithm.

4. MATHEMATICAL MODELS AND SOLUTION METHOD FOR UMDCP$_2$

Our second uncapacitated minimum distribution cost problem, UMDCP$_2$, finds the minimum cost to receive a unit flow for the requested T-node from an S-node. Unlike UMDCP$_1$ which only considers the dependency relationship between arcs, UMDCP$_2$ sends flows from the S-node to the T-node considering the flow value for each arc and the distribution ratios of D-nodes.

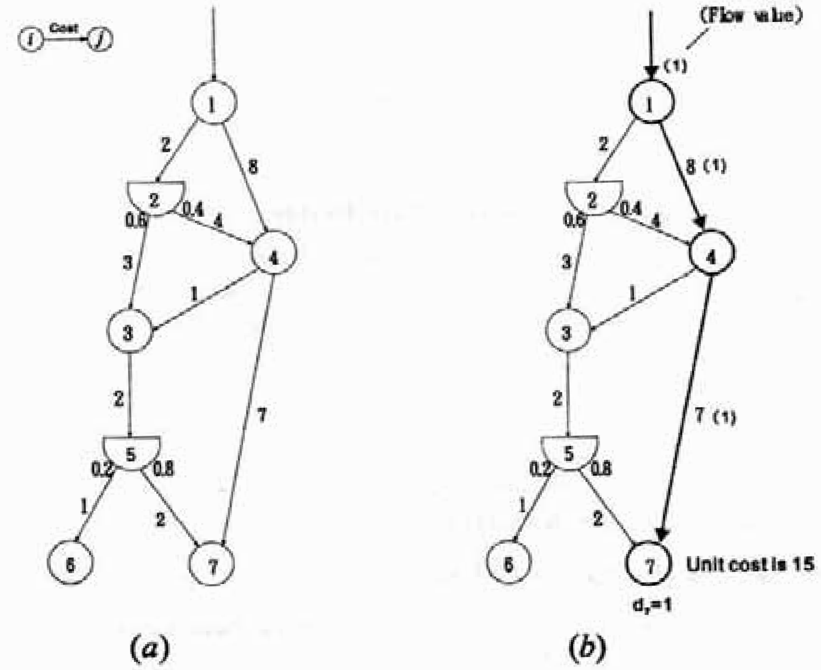


Figure 4: A UMDCP$_2$ Example

For the one-to-one UMDCP$_2$, once a D-node appears in a main path connecting the source s and sink t, all of its outgoing arcs have to be selected. However, among these outgoing arcs, only one will be in the main path connecting s and t. Figure 4(a) shows a UMDCP$_2$ example with its solution illustrated in Figure 4(b).

To formulate the one-to-one UMDCP$_2$, we construct a transformed network G' by adding a parallel artificial arc $(i, j)'$ and set its length to be c_{ij} for each arc (i, j) out of a D-node i. Like UMDCP$_2$, when a D-node i lies on a main path connecting the requested source and sink nodes, only one of its outgoing arcs will appear in the main path. In that case, we select that arc together with all other parallel arcs into the solution. The flows on these selected arcs can be calculated according to the distribution ratios. For each sink node i in N_T we associate it with a demand $d(i) > 0$ where $d(i) \geq 1$ if i is the requested sink node. Let B be the set of all arc $(i, j)'$ for each $i \in N_D$ and for each $j \in L(i)$.

Unlike UMDCP$_1$, here we must consider the flow value x_{ij} for each arc (i, j) and the distribution ratios r_{ik} for each outgoing arc (i, k) of each D-node i. We associate a binary variable y_{ij} to each arc (i, j) in the transformed network to denote whether the flow passes arc (i, j) (i.e., $y_{ij} = 1$) or not (i.e., $y_{ij} = 0$). Suppose M is a very large number, we can formulate the one-to-one UMDCP$_2$ as follows:

$$\min \textstyle\sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{(i,j)\in B} c_{ij}x_{ij} \qquad (\text{UMDCP}_2)$$

$$s.\ t. \quad \textstyle\sum_{k\in L(i)} x_{ik} \sum_{j\in E(i)} x_{ji} = 0 \ \forall i \in N_O \qquad (10)$$

$$\textstyle\sum_{j\in E(i)} x_{ji} \geq d(i) \ \forall i \in N_T \qquad (11)$$

$$r_{ik}x_{ji} - x_{ik} - x'_{ik} = 0 \quad \forall j \in E(i), \forall k \in L(i), \forall i \in N_D \qquad (12)$$

$$x_{ji} - My_{ji} \leq 0 \ \forall j \in E(i) \ \forall i \in N_D \qquad (13)$$

$$x_{ik} - My_{ik} \leq 0 \ \forall k \in L(i) \ \forall i \in N_D \qquad (14)$$

$$x'_{ik} - My'_{ik} \leq 0 \ \forall k \in L(i) \ \forall i \in N_D \qquad (15)$$

$$\textstyle\sum_{(i,k)\in L(i)} y_{ik} \leq 1 \ \forall i \in N_D \qquad (16)$$

$$y_{ij}, y'_{ij} \in \{0, 1\} \ \forall (i, j) \in A \text{ and } \forall (i, j) \in B \qquad (17)$$

The objective function minimizes the total cost for all arcs in G', including those parallel arcs in the network. Equation (10) is the flow balance constraint for all O-nodes. Equation (11) shows that all T-nodes must satisfy their demand. Equation (12) represents the flows on D-nodes' outgoing arcs should obey the distribution ratios. Equations

(13), (14) and (15) stand for the relationships between the flow values and whether the flow passes an arc or not. That is, an arc (i, j) is selected ($y_{ij} = 1$) if $x_{ij} > 0$. Otherwise, arc (i, j) is not selected ($y_{ij} = 0$) if $x_{ij} = 0$. Equations (16) and (17) are relative to D-nodes: only one of the original arc and its parallel arc can be selected, and if an original arc is selected, then all other parallel arcs from the same D-node will also be selected. $y_{ij} = 1$ or $y'_{ij} = 1$ means arc (i, j) is selected, otherwise $y_{ij} = y'_{ij} = 0$.

Modified Dijkstra's Algorithm for one-to-one UMDCP$_2$

begin
$S := \{s\}$; $\bar{S} := N - S$;
$d(s) := 0$; $avg\_d(s) := 0$; $x_{sj} := \beta \ \forall (s, j) \in A(s)$;
$d(i) := \infty$, $avg\_d(i) := 0 \ \forall i \in \bar{S}$
pred(i) $:= 0 \ \forall i \in N$;
while $S \neq N$ do
select the node $i = \arg\min_j \{avg\_d(j) : j \in \bar{S}\}$;
$S := S \cup \{i\}$; $\bar{S} := \bar{S} - \{i\}$;
if $i \in N_O$ then
for each arc $(i, j) \in A(i)$ do
$x_j := x_{ij}$;
$x_{jk} := r_{jk} x_{ij}$, $x_j := x_{ij} \ \forall (j, k) \in A(j)$;
if $j \in N_D$ then
$d(j) := d(i) + c_{ij} + \sum_{(j,k) \in A(j)} c_{jk}$;
$avg\_d(j) := d(j)/x_j$; pred $(j) := i$;
for each arc $(j, k) \in A(j)$ do
if $avg\_d(k) > d(k)/x_k$ then
$d(k) := d(j)$;
$avg\_d(k) := d(k)/x_k$; pred$(k) := j$;
else
if $avg\_d(j) > (d(i) + c_{ij} x_{ij})/x_j$ then
$d(j) := d(i) + c_{ij} x_{ij}$
$avg\_d(j) := (d(i) + c_{ij} x_{ij})/x_j$; pred $(j) := i$;
end

Figure 5: A Modified Dijkstra's Algorithm for Solving a One-to-one UMDCP$_2$.

Similar to UMDCP$_1$, the 0-1 integer programming model for UMDCP$_2$ may potentially require exponential time to solve. With careful analysis, here we also present a modified Dijkstra's algorithm to solve a UMDCP$_2$ in polynomial time.

Denote the amount of flow passing through each node $i \in N$ to be x_i. Let $d(i)$ and $avg\_d(i)$ (equals to $d(i)/x_i$) be the distance and the average distance from node S to node i for $i \in N$. We give an algorithm in Figure 5 similar to the one in Figure 2 to solve a one-to-one UMDCP$_2$. In particular, the algorithm selects a node according to its average distance label $avg\_d(i)$, instead of the original distance label $d(i)$. Whenever a D-node is encountered, the algorithm takes all of its outgoing arcs into account. Using the predecessor information, the optimal solution can be retrieved by the main path connecting from s to t, together with all the outgoing arcs of the D-nodes in the main path.

The complexity of our one-to-one UMDCP$_2$ algorithm is also the same as the conventional Dijkstra's algorithm (see Section 3.1).

5. PREPROCESS

We may transform a distribution network into an equivalent one of smaller size by removing some nodes or arcs, or integrating some arcs. There are three cases of compacting a distribution network: Case 1 compacts O-nodes, Case 2 compacts D-nodes, and Case 3 compacts parallel arcs.

Case 1: Any *O*-node with Only One Incoming Arc and One Outgoing Arc can be Compacted

When an *O*-node has a single incoming arc and a single outgoing arc, this *O*-node can be regarded as a transshipment node. Thus, we can remove this *O*-node and merge its adjacent arcs into one arc with a new cost equals to the summation of the costs on the two original arcs.

Take the distribution network in Figure 6(a) for example. After removing node 4 and merging arc (2,4) and (4, 5), we have arc (2,5) with cost as 3 + 2 = 5 shown in Figure 5(b). After this compacting procedure, one *O*-node and one arc can be removed each time. Hence, Case 1 will be executed $O(n)$ times.

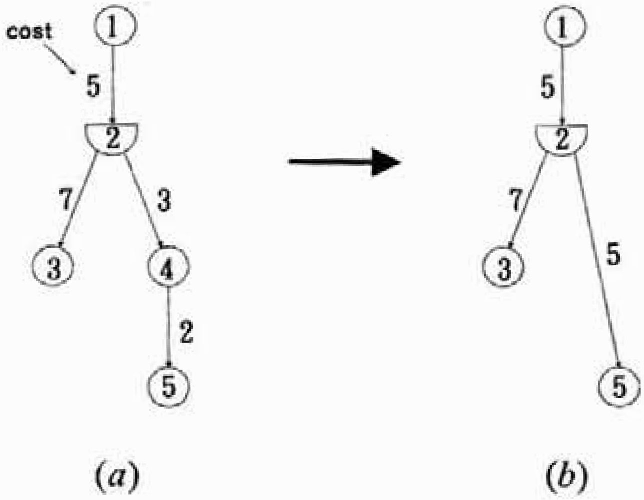


Figure 6: Any *O*-node with Only one Incoming Arc and One Outgoing Arc can be Compacted

Case 2: Nodes in the Same D-group can be Compacted Into a *D*-node

A *D*-group is a set of adjacent *D*-nodes. Depending on the problem characteristics, we discuss two subcases: (1) Case 2.1 for UMDCP$_1$, which only considers the flow dependency relation but not the distribution ratios and arc flow; and (2) Case 2.2 for UMDCP$_2$, which considers the distribution ratios of a *D*-node and arc flows.

Case 2.1: In UMDCP$_2$, we can merge all adjacent *D*-nodes to be the top *D*-node closest to the source node. Then we add new arcs from this *D*-node to all the *O*-nodes adjacent to the *D*-group. The compacting procedures are as follows:

Step 0: Identify a *D*-group $\bar{G}$ which contains several adjacent *D*-nodes. Suppose its top *D*-node j_0 is connected from an *O*-node i_0. Repeat Step 1 to Step 3 for each $\bar{G}$.

Step 1: Identify all the *O*-nodes (say, $v_1, \dots v_\chi$) adjacent to this *D*-group. For each arc (j_a, j_b) that connects *D*-nodes j_a and j_b in $\bar{G}$, we count the number (say, n_a) of *O*-nodes reachable from j_a, then we change the arc length $c_{j_a j_b}$ to be $c_{j_a j_b}/n_a$.

Step 2: For each *O*-node v_i other than i_0 that is adjacent to this *D*-group, add a new arc (j_0, v_i) and associate it with a length $c_{j_0 v_i} = \sum_{(u,v) \text{ in the path } j_0 \to v_i} c_{uv}$ where c_{uv} is the new arc length as calculated in Step 1.

Step 3: Retain $i_0, j_0, (i_0, j_0)$ and all the new arcs (k_0, v_i), remove all the other arcs connecting to or within $\bar{G}$.

We take Figure 7(a) for example. The *D*-group is the set of node 2, node 4 and node 6. Retain the top *D*-node 2, arc (1, 2) and (2, 3). We know that node 4 can reach three *O*-nodes through *D*-group and node 6 can reach two *O*-nodes, so we change c_{24} to be $c_{24}/3$ and c_{46} to be $c_{46}/2$ as shown in Figure 7(b). Add three new arcs (2, 5), (2, 7) and (2,8) and associate them with a length equal to the length of 2– 4 – 5, 2 – 4 – 6 – 7, and 2 – 4 – 6 – 8, respectively. Hence, the costs of arcs (2, 5), (2, 7) and (2, 8) in Figure 7(c) are 1 + 2 = 3, 1 + 2 + 2 = 5 and 1 + 2 + 5 = 8.

Case 2.2: In our UMDCP$_2$, we can also merge all adjacent *D*-nodes to the top *D*-node. It differs from case 2.1 in consideration of the distribution ratios of each *D*-node. The compacting procedures are as follows:

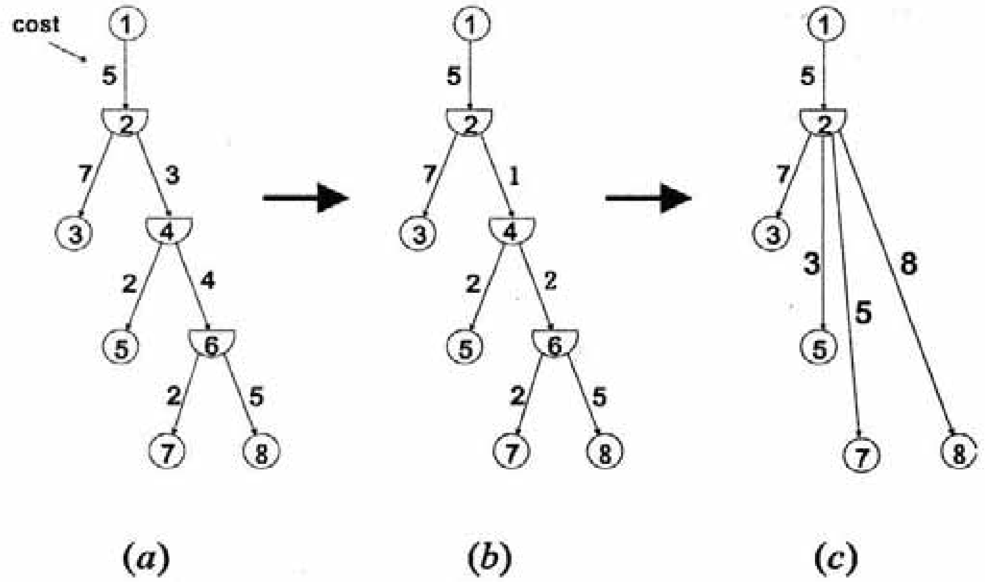


Figure 7: Adjacent *D*-nodes can be Compacted into One D-node in UMDCP$_1$

Step 0: Identify a *D*-group $\bar{G}$ which contains several adjacent *D*-nodes. Suppose its top *D*-node j_0 is connected from an *O*-node i_0. Repeat Step 1 to Step 3 for each $\bar{G}$.

Step 1: Identify all the *O*-nodes (say, $v_1, \ldots v_\chi$) adjacent to this *D*-group.

Step 2: For each *O*-node v_i other than i_0 that is adjacent to this *D*-group, add a new arc $(j_0 v_i)$ and associate it with a length $c_{j_0 v_i} = \Sigma_{(u,v) \text{ in the path } j_0 \to v_i} c_{uv}$ and distillation ratio $k_{j_0 v_i} = \Pi_{(u,v) \text{ in the path } j_0 \to v_i} k_{uv}$.

Step 3: Retain $i_0, j_0, (i_0, j_0)$ and all the new arcs (j_0, v_i), remove all the other arcs connecting to or within $\bar{G}$.

For example, in Figure 7(a), we have three new arcs (2.5), (2,7), and (2, 8). Retain the top *D*-node 2, arc (1, 2) and (2, 3). Then $c_{25} = 3 + 2 = 5$ and $k_{25} = 0.8.\ 0.4 = 0.32$; $c_{27} = 3 + 4 + 2 = 9$ and $k_{27} = 0.8 \cdot 0.6 \cdot 0.3 = 0.144$; $c_{28} = 3 + 4 + 5 = 12$ and $k_{28} = 0.8 \cdot 0.6 \cdot 0.7 = 0.336$.

In Case 2, one *D*-node and one arc can be reduced at least each time. So, Case 2 will be executed $O(m)$ times.

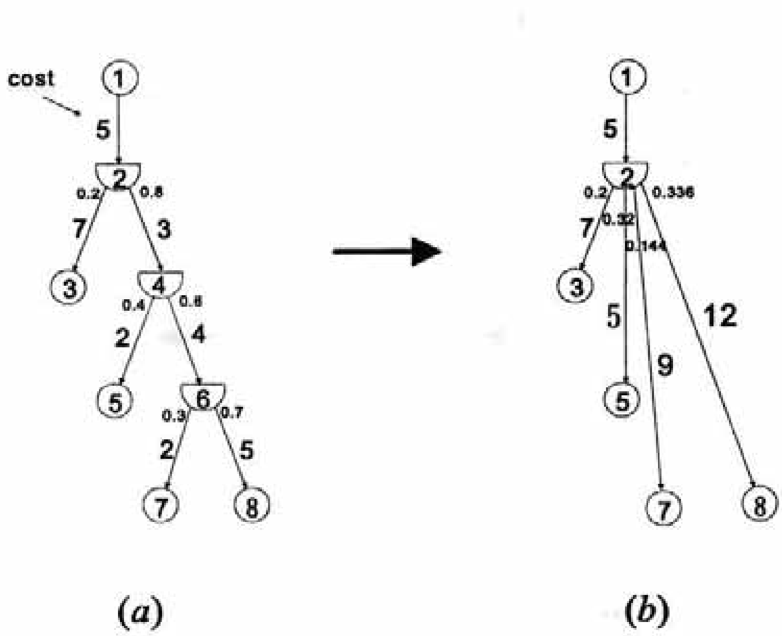


Figure 8: Adjacent *D*-nodes can be Compacted into One *D*-node in UMDCP$_2$

Case 3: Parallel Arcs Connecting to the Same End Nodes can be Compacted

Parallel arcs are arcs that have the same head and tail nodes but may have different arc lengths. We can merge them into one new arc from the tail node to the head node, and sum up their costs to be the cost on the new merged arc. See Figure 9, Figure 10 and Figure 11 for examples. One arc can be reduced each time in Case 3, thus Case 3 will be executed $O(m)$ times.

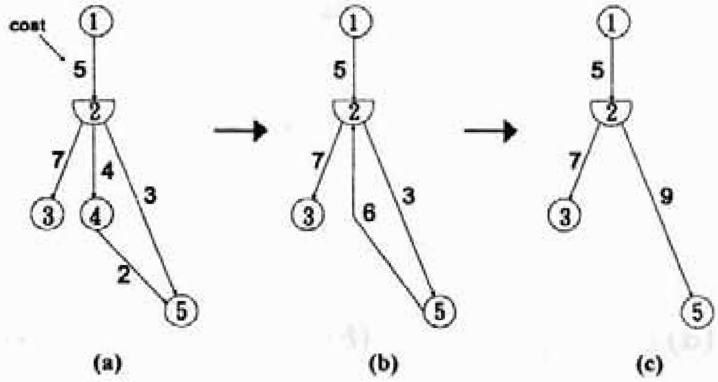


Figure 9: Parallel Arcs Connecting to the Same Nodes can be Compacted (1)

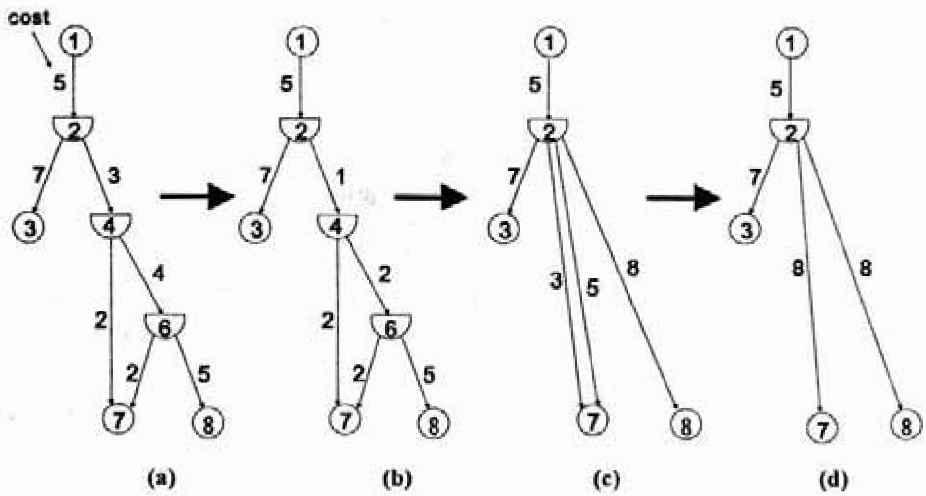


Figure 10: Parallel Arcs Connecting to the Same Nodes can be Compacted (2)

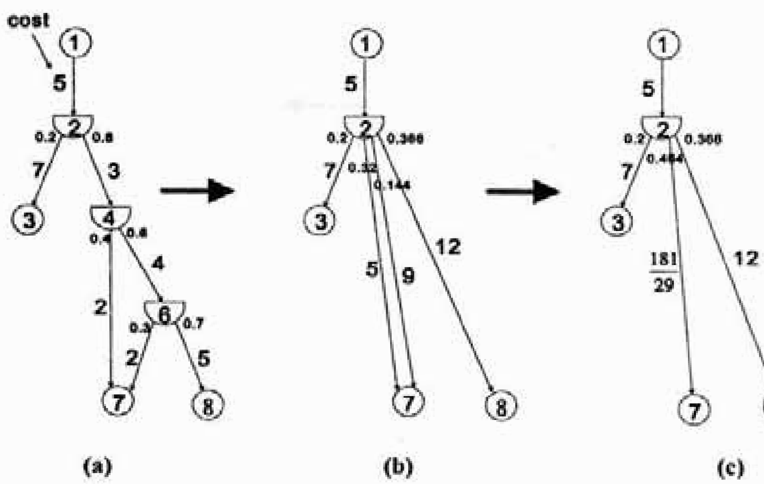


Figure 11: Parallel Arcs Connecting to the Same Nodes can be Compacted (3)

It can be shown that these 3 cases may induce each other. Each compaction reduces at least one arc or node and the number of nodes and arcs is finite, so the entire compacting procedures terminates in $O(\max\{m, n\})$ iterations, and thus in polynomial time since each compacting case takes polynomial time. In summary, for any distribution network in our UMDCP problems, we may compact it to obtain an equivalent network of smaller size. There are two properties for a compacted distribution network. First, each intermediate node (an *O*-node or a *D*-node) has at least three arcs, including one incoming arc and one outgoing arc. Second, no *D*-group exists in a compacted distribution network.

4. CONCLUSIONS AND FUTURE RESEARCH

Extending the MNF model presented by Fang and Qi [5], this paper introduces two uncapacitated minimum distribution cost problems (UMDCP), proposes their formulations and develops polynomial-time solution methods based on shortest path algorithms. Since all of our UMDCPs have no arc capacity constraints, we treat them as specialized shortest path problems with side constraints called the flow distillation constraints which specify the distilling relationship for the flows entering and leaving any *D*-node.

The $UMDCP_1$ finds the minimum cost to connect an *S*-node to a *T*-node. When a *D*-node is selected to be in the main path in a solution, all of its outgoing arcs also have to be in the solution. For its one-to-one case, among all the outgoing arcs from a selected *D*-node, only one can further connect to other arcs. In $UMDCP_1$, the cost is related to whether an arc is in a solution or not, and we ignore the cost of flows. We have proposed two mathematical formulations for the one-to-one $UMDCP_1$, transform it to be a conventional shortest path problem and solve it by Dijkstra's algorithm.

The $UMDCP_2$ finds the minimum cost to receive a unit flow the requested *T*-node from the *S*-node. The solution for $UMDCP_2$ is a subgraph that contains paths connecting the requested *S*-node and *T*-nodes with some side branch arcs. Similar to the $UMDCP_1$, the $UMDCP_2$ also allows only one of the arcs outgoing from a selected *D*-node to connect further to other arcs in its one-to-one case. However, unlike $UMDCP_1$ that considers only the dependency relations between arcs, $UMDCP_2$ focuses on the total unit flow costs to be received for its requested destination. We have proposed a mathematical formulation, give a modified Dijkstra's algorithm to solve the one-to-one $UMDCP_2$.

In this paper, we only considered compacted distribution networks since they are equivalent to the original one but with smaller size. We propose the compacting rules and give properties for a compacted network. Such a preprocessing operation will simplify the network and offers better managerial insights.

There are still many challenging problems related to the topics discussed in this paper. For example, how to give an efficient graphical algorithm to solve the original UMDCP is still unknown. Also, for the one-to-some $UMDCP_1$ and $UMDCP_2$ cases, we are still not able to develop any efficient graphical algorithms. Another interesting topic would be to consider the combinational node (*C*-node) and inventory node (*I*-node) introduced in Fang and Qi [5] which makes the model more realistic.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti and J. B. Orlin (1993), Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, New Jersey, USA.

[2] R. K. Ahuja, K. Mehlhorn, J. B. Orlin, R. E. Tarjan (1990), Faster Algorithms for the Shortest Path Problem. *Journal of ACM*; 37: 213–223.

[3] A. R. Dial (1969), Algorithm 360: Shortest Path Forest with Topological Ordering. *Communications of the ACM* **12:** 632–633.

[4] R. E. W. Dijkstra (1959), A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik* **1:** 269–271.

[5] S. C. Fang, L. Qi (2003), Manufacturing Network Flows: A General Network Flow Model for Manufacturing Process Modeling. *Optimization Methods and Software* **18:** 143–165.

[6] M. L. Fredman, R. E. Tarjan (1987), Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Journal of the ACM* **34:** 596–615.

[7] E. L. Johnson (1972), On Shortest Paths and Sorting. In Proceedings of the ACM 25th Annual Conference 510–517.