國　立　成　功　大　學

工 業 與 資 訊 管 理 研 究 所

碩　士　論　文

Uncapacitated Minimum Cost Flow Problem in a Distribution Network

指 導 教 授：王 逸 琳　博士

研　究　生　：楊 羽 惠

中 華 民 國 九 十 四 年 七 月

國立成功大學
碩士論文

Uncapacitated Minimum Cost Flow Problem in a Distribution Network

研究生：楊羽惠

本論文業經審查及口試合格特此證明

論文考試委員：

陳 文 智	陳文智
許 瑞 麟	許瑞麟
李 宇 欣	李宇欣
王 逸 琳	王逸琳

指導教授： 王 逸 琳
系(所)主管： 王 泰 裕

中華民國 九十四 年 五 月 二十九 日

博碩士論文授權書

（國科會科學技術資料中心版本 93.2.6）

本授權書所授權之論文為本人在 國立成功大學 大學（學院）工業與資訊管理 系所 ＿＿＿＿ 組 九十三 學年度第 二 學期取得 碩 士學位之論文。

論文名稱：Uncapacitated Minimum Cost Flow Problem in a Distribution Network

☑同意　☐不同意

本人具有著作財產權之論文全文資料，授予行政院國家科學委員會科學技術資料中心（或其改制後之機構）、國家圖書館及本人畢業學校圖書館，得不限地域、時間與次數以微縮、光碟或數位化等各種方式重製後散布發行或上載網路。

本論文為本人向經濟部智慧財產局申請專利（未申請者本條款請不予理會）的附件之一，申請文號為：＿＿＿＿＿＿，註明文號者請將全文資料延後半年再公開。

☑同意　☐不同意

本人具有著作財產權之論文全文資料，授予教育部指定送繳之圖書館及本人畢業學校圖書館，為學術研究之目的以各種方法重製，或為上述目的再授權他人以各種方法重製，不限地域與時間，惟每人以一份為限。

上述授權內容均無須訂立讓與及授權契約書。依本授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。上述同意與不同意之欄位若未鉤選，本人同意視同授權。

指導教授姓名：王逸琳

研究生簽名：　　　　　　　　　　學號：R36921158

（親筆正楷）　　　　　　　　　　（務必填寫）

日期：民國 94 年 7 月 23 日

1. 本授權書（得自http://sticnet.stic.gov.tw/sticweb/html/theses/authorize.html 下載或至http://www.stic.gov.tw首頁右下方下載）請以黑筆撰寫並影印裝訂於書名頁之次頁。
2. 授權第一項者，請確認學校是否代收，若無者，請自行寄論文一本至台北市（106）和平東路二段 106 號 1702 室 國科會科學技術資料中心 黃善平小姐。（本授權書諮詢電話：02-27377606 傳真：02-27377689）

誌謝

本篇論文於撰寫期間，承蒙指導教授王逸琳老師的勉勵及指導，方能順利完成。王老師無論是在專業知識上的素養，亦或待人處事方面，及對學生的用心與付出，一直都是我努力學習的榜樣。口試期間，承蒙許瑞麟老師、李宇欣老師及陳文智老師的詳加指正與建議，使本篇論文更加完備，在此謹獻上最誠摯的感謝。

研究所碩士班兩年的時光轉眼即逝，非常慶幸有緣能與班上所有的同學及學弟妹們相互切磋砥礪，尤其是陪伴我經歷歡笑與淚水、給予我最大鼓勵及幫助的好朋友們，在此無法一一列出，但內心依舊感激。如今，欣喜彼此都能順利畢業，踏入人生的另一個階段，互道珍重之餘，願人生道路上仍充滿更多喜悅與祝福。

我也要感謝我的家人，感謝爺爺奶奶對我無微不至的照顧、感謝爸爸媽媽對我的栽培及鼓勵、感謝弟妹在我失落時的加油打氣，若沒有家人的支持，就不會有今天的我，此情永銘我心。

最後，我還要謝謝我的男朋友—威漢，在我求學遇到挫折困境時，總是不斷地體諒並指引我，他對我的關懷與鼓勵，是我走下去的最大動力。

沒有大家的包容與厚愛，論文可能無法完成，對於大家的關懷無以言謝，僅能以更認真的態度貢獻所學，迎接未來的挑戰。

楊羽惠　謹誌於台南

九十四年七月

ABSTRACT

Uncapacitated minimum cost flow problems in a distribution network

by
Yu-Hui Yang

In this thesis, we consider three special minimum cost flow problems in a distribution network, a kind of manufacturing network recently introduced by Fang and Qi. The network differs from a traditional network model because 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. Besides, all the arcs in our models have no upper bounds. We treat these uncapacitated minimum cost flow problems as specialized shortest path problems with side constraints in a distribution network. We give a preprocessing that compacts a distribution network to an equivalent network of smaller size, derive their mathematical formulations and develop efficient solution methods.

Keywords : Manufacturing network, distribution network, minimum cost flow problem, uncapacitated, shortest path

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER I

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 modelling 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 (2003) presented a generalized network model called the *manufacturing network flow* (MNF) for this purpose. In such a model, multiple products may be produced by multiple raw-materials via a combination of synthesis and distilling operation.

1.1 Motivation

We investigate three different problems which contain extra working stations, called D-nodes here, in the industrial manufacturing regions. In all the three cases, the manufacturing process may start and end at one or several working stations (nodes). Since all three cases do not consider capacity constraints associated with nodes or arcs, we call our problems as Uncapacitated Minimum Distribution Cost Problem, denoted by UMDCP$_i$ where $i = 1, 2, 3$ corresponding to our three cases, respectively.

The first case (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 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 an extra working station (i.e. a D-node), it has to produce several different products according to some predefined fixed ratios. Among these products made by a D-node, only one product will be processed further. This may happen when it is worth to outsource other products which are processed from a D-node. Sometimes this may be a result of some contract which only allows one out of many products processed from a D-node to be further processed. The goal is to minimize the total fixed cost required to produce the set of requested products.

Instead of considering the fixed cost, the second and third case consider the unit cost and the flows in a manufacturing network. Like the first case, once a material enters a D-node, among those products made by a D-node, the second case (UMDCP$_2$) only allows one product to be processed further. The objective is to seek the minimum total unit costs to produce the set of requested products.

The third case (UMDCP$_3$) is the most general case and has been mentioned by Fang and Qi (2003). Unlike UMDCP$_2$ which only allows one out of many products produced by a D-node, UMDCP$_3$ let all products produced by a D-node keep processing until the set of requested products are produced with minimum total unit costs.

In this thesis, we are concerned with the minimum cost flow problem in a distribution network, a kind of manufacturing network recently introduced by Fang and Qi (2003). It differs from a traditional network model because 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. In their model, the flow for each arc

(i, j) in the arc set A has a lower bound $l_{ij} = 0$, but no upper bound (i.e., $u_{ij} = \infty$). Each arc in the model has its own unit cost and flow value, denoted by c_{ij} and x_{ij}, respectively. The flow value for each arc satisfies the boundary constraint, $0 \le x_{ij} \le \infty$ for each arc $(i, j) \in A$. In a traditional minimum cost flow problem, the flow value has a lower bound and an upper bound. Generally, the lower bound is 0. When there is no upper bound, i.e., $u_{ij} = \infty$ for each arc $(i, j) \in A$, a traditional min-cost flow problem can be viewed as a shortest path problem. Fang and Qi (2003) propose a modified network simplex algorithm to solve the uncapacitated minimum distribution cost problem. Since this problem can be viewed as a shortest path problem by this observation, we will solve it by modified shortest path algorithms.

1.2 Objectives

Intrigued by the uncapacitated minimum distribution cost problem of Fang and Qi (2003), here we extend this topic and propose three uncapacitated minimum distribution cost problems: UMDCP$_1$, UMDCP$_2$, and UMDCP$_3$. These problems can be viewed as specialized shortest path problems to some extent.

Our first problem, UMDCP$_1$, finds the minimum cost to connect a source node (S-node) to several termination nodes (T-nodes). In this problem, we only consider the dependency of arcs connecting to a D-node and ignore its flow distillation constraints. In particular, given a MNF graph, we start to select arcs to form a subgraph which connects the S-node and T-nodes with minimum total cost which is calculated by summing up all the cost associated with the selected arcs. In this case, the solution is a subgraph that contains paths connecting S-node and all the requested T-nodes, together with some side branch arcs that are side-products of some D-nodes selected in the paths. This problem can be formulated as a 0, 1 integer programming problem in which we set a binary variable z_{ij} for each arc (i,j) to represent whether arc (i,j) can be selected (i.e., $z_{ij}=1$) or not (i.e., $z_{ij}=0$). Note that once the incoming arc for a D-node is selected, all of its outgoing arcs must also be selected. The objective thus becomes to find min $\sum\limits_{(i,j)\in A} c_{ij}z_{ij}$ for each arc (i,j) such that the requested S-node and T-nodes are connected. Chapter 3 discusses details of this problem.

Our second problem, UMDCP$_2$, finds the minimum cost to receive a unit flow for each requested T-nodes from an S-node. Like UMDCP$_1$, the solution is a subgraph that contains paths connecting the requested S-node and T-nodes with some side branch arcs. Unlike UMDCP$_1$ that considers only the dependency relationship between arcs, UMDCP$_2$ further considers the flow distillation constraints for each D-node, and allows the flow to be stored on any O-node. The flow distillation constraints complicates this problem. This problem can be formulated as a mixed integer linear programming problem (MIP) as will be illustrated in Chapter 4.

Our last problem, UMDCP$_3$, also finds the minimum cost to receive a unit flow for each requested T-node from an S-node as UMDCP$_2$. However, unlike UMDCP$_2$, this problem does not allow flow to be stored at any node. That is, all the flows must be sent to the requested T-nodes or some other T-nodes. This problem can be formulated as a linear programming problem (LP) and has been investigated by Fang and Qi (2003) by a modified network simplex algorithm. We will discuss more details of this problem in Chapter 5.

Since we view the same problem as a shortest path problem, our goal here is to propose more efficient algorithms to solve the same problem.

1.3 Thesis outline

The remainder of this thesis is organized as follows. Literature review is presented in Chapter 2 that introduces the background for our research. In Chapter 3 and 4, we define three specialized shortest path problems, give formulations and solution methods for the one-to-one and one-to-some cases. Chapter 5 is concluded by summarizing our results, listing our contributions and proposes the future research directions.

CHAPTER II

LITERATURE REVIEW

This Chapter is partitioned into five Sections. In Section 2.1, we define the notations for a general network. Section 2.2 introduces a MNF problem—minimum cost flow problem, and two of its special cases: minimum distribution cost problem and minimum assembly cost problem. Section 2.3 reviews the traditional shortest path problems and algorithms. Section 2.4 illustrates Dijkstra's Algorithm. A brief summary is concluded in Section 2.5.

2.1 Notations

We adopt the notations of the paper by Fang and Qi (2003). Let $G = (N, A)$ be a general network, and denote N and A to be the node set and arc set. The numbers of nodes and arcs are n and m. We say i is the tail node and j is the head node for an arc $(i, j) \in A$. We also denote x_{ij} to be the flow value on a specific arc (i, j) with a given upper bound $u_{ij} > 0$ satisfying the capacity constraint, i.e.,

$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A(2.1)$$

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 all "entering" nodes to node i and the set of all "leaving" nodes from node i as

$$E(i) := \{j \in N : (j,i) \in A\}$$

$$L(i) := \{j \in N : (i,j) \in A\}$$

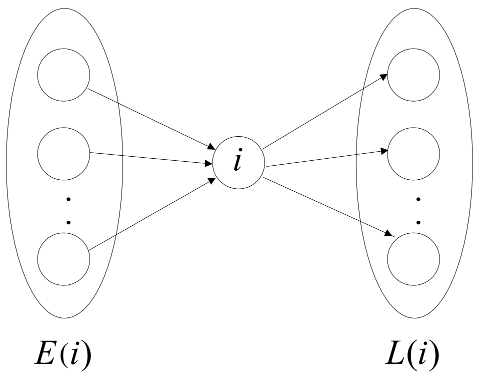

Figure 2.1: $E(i)$ and $L(i)$

There are six classes of nodes in a generalized network model.

1. *O*-**nodes** are ordinary nodes for transition. These nodes may have several incoming arcs and outgoing arcs, but only one kind of material/semi-product can flow through it with balance, i.e.,

$$\sum_{j \in E(i)} x_{ji} = \sum_{j \in L(i)} x_{ij} \quad \forall i \in N_O (2.2)$$

where N_O represents the set of all O-nodes.

2. *S*-**nodes** are source nodes for raw materials. It may have one or several source nodes in a manufacturing network, and each of them represents one different raw-material. For each $i \in N_S$, $E(i) = \emptyset$ and we associate it with an input flow of x_i such that

$$x_i = \sum_{j \in L(i)} x_{ij} \quad \forall i \in N_S (2.3)$$

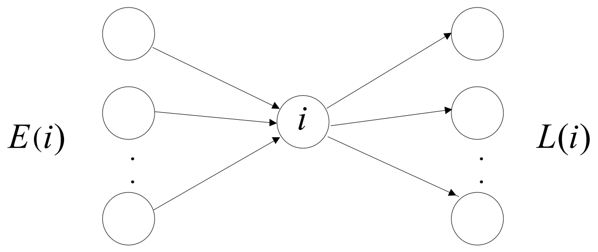

Figure 2.2: An O-node

where N_S represents the set of all S-nodes. Note that only one kind of raw-material is allowed at an S-node.

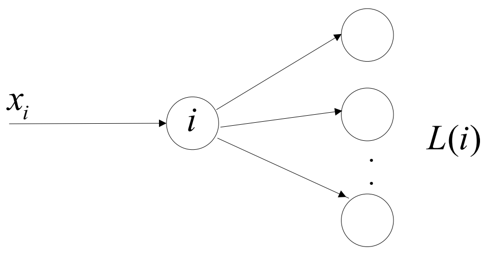

Figure 2.3: An S-node

3. T**-nodes** are termination nodes for final products. It may have one or several termination nodes in a manufacturing network, and every node represents one different final-product. For each $i \in N_T$, $L(i) = \emptyset$ and there associates an output flow of x_i such that

$$x_i = \sum_{j \in E(i)} x_{ji} \quad \forall i \in N_T (2.4)$$

where N_T represents the set of all T-nodes. Note that only one kind of final product is allowed to flow in a T-node.

4. I**-nodes** are inventory nodes for in-process materials/semi-products. I-nodes are like O-nodes, except the total flow value of the incoming arcs is higher than the

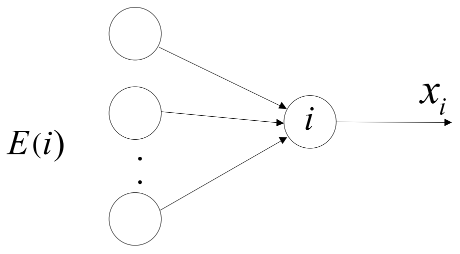

Figure 2.4: A T-node

total flow value of the outgoing arcs, i.e.,

$$\sum_{j \in E(i)} x_{ji} = x_i + \sum_{j \in L(i)} x_{ij} \quad \text{and} \quad x_i \geq 0 \quad \forall i \in N_I (2.5)$$

where N_I represents the set of all I-nodes and x_i is the excess quantity. For an I-node, only one kind of in-process material/semi-product is allowed to flow through, with an excess quantity x_i stored in the node.

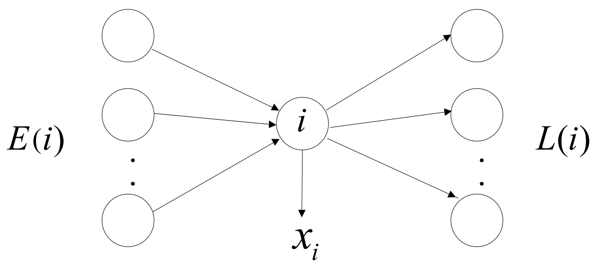

Figure 2.5: An I-node

5. D**-nodes** are distillation nodes. A D-node 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. For each $i \in N_D$, we have

$$E(i) = \{i^*\}$$

and

$$x_{ij} = k_{ij} x_{i^* i} \quad \forall j \in L(i)(2.6)$$

where N_D represents the set of all D-nodes and k_{ij} is a positive real number. We call this as a *flow distillation* constraint. For example, a distillation operation that decomposes one material into several products with fixed ratios.

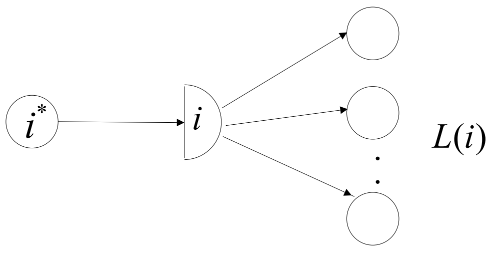

Figure 2.6: A D-node

6. C**-nodes** are combination nodes. A C-node has only one outgoing arc (for one kind of product) but multiple incoming arcs (for different kinds of materials), and the flow value on an incoming arc is proportional to the flow value on the outgoing arc. For each $i \in N_C$, we have

$$L(i) = \{i^*\}$$

and

$$x_{ji} = h_{ji}x_{ii^*} \quad \forall j \in E(i)(2.7)$$

where N_C represents the set of all C-nodes and h_{ji} is a positive real number. For example, a synthesis unit that assembles several materials with fixed ratios to form a new product.

In this thesis, we consider a manufacturing network flow model (MNF) in which there are four kinds of nodes: O-nodes, S-nodes, T-nodes and D-nodes.

2.2 General MNF Optimization Problems

Given a manufacturing network $G = (N, A)$, a feasible flow x is a mapping from A to $\Re^+$ such that it satisfies the constraints (2.1-2.7) of the general network. Let F

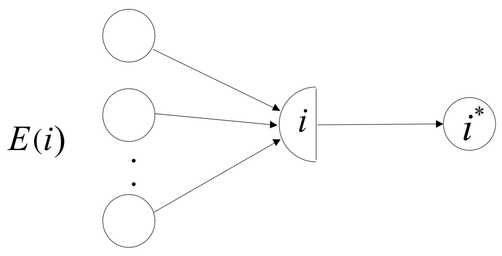

Figure 2.7: A C-node

be the set of feasible flows. Fang and Qi (2003) present a general MNF optimization problem shown as follows:

Assume that there is a market demand $d_j > 0$ for each final product j, and a unit cost c_i for each raw-material. The minimum cost flow problem finds a feasible flow that minimizes the total cost of raw-materials, i.e.,

$$\begin{array}{lll} \min & \sum_{i \in N_S} c_i x_i & \\ \textit{subject to} & x \in F & (2.8) \\ & x_j \geq d_j \quad \forall j \in N_T & \end{array}$$

This problem is a linear programming problem which has an embedded network flow structure with side constraints.

Fang and Qi (2003) discuss two special cases—minimum distribution cost problem and minimum assembly cost problem are presented.

A. Minimum distribution cost problem. Consider a general supplier who wants to distribute certain amount of a particular product from S-node s to several retailers at different T-nodes in a distribution network where O-nodes are used for transition and D-nodes are used for proportional distribution at middle stages. For simplicity, we assume that

$$\sum_{j \in L(i)} k_{ij} = 1 \quad \forall i \in N_D (2.9)$$

Since no C-nodes and I-nodes are involved, we let F_D be the set of all flows that satisfy the constraints (2.1-2.4), (2.6), and (2.9). Then we have

$$\begin{array}{ll} \min & c_s x_s + \sum_{(i,j)\in A} c_{ij} x_{ij} \\ subject\ \ to & x \in F_D \\ & x_j \geq d_j \quad \forall j \in N_T \\ & x_s \leq u_s \end{array} \tag{2.10}$$

where c_s is the unit cost of the raw-material and c_{ij} is the cost of each unit flow going through an arc (i, j).

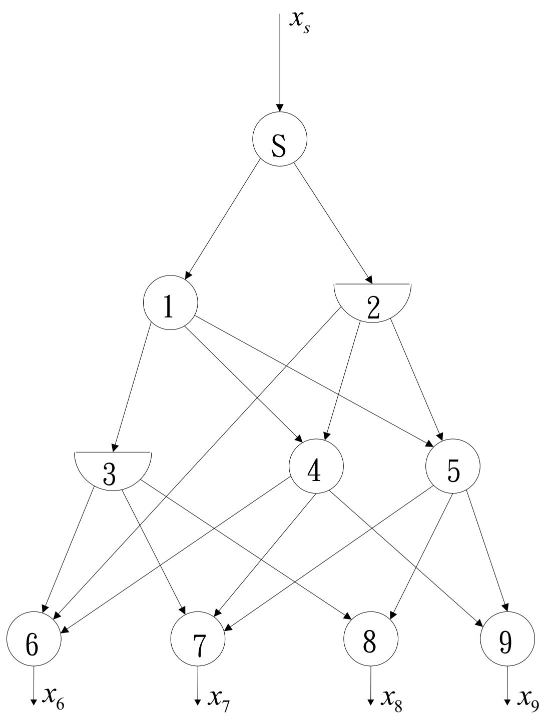

Figure 2.8: A Distribution Network

B. Minimum assembly cost problem. An assembly network looks like a reversed distribution network. Consider a manufacturer who wants to produce one final product at the termination node t by assembling various components obtained at different source nodes $i \in N_S$ in a manufacturing network where O-nodes satisfy the flow balance equation and different kinds of parts are assembled at C-nodes to form a semi-product/product according to fixed ratios. We let F_C be the set of all flows that satisfy the constraints (2.1-2.5), (2.7). Then we have

$$\begin{array}{lll} \min & \sum_{i \in N_S} c_i x_i + \sum_{(i,j) \in A} c_{ij} x_{ij} & \\ \textit{subject to} & x \in F_C & \\ & x_i \le u_i \quad \forall i \in N_S & (2.11) \\ & x_t \ge d_t & \end{array}$$

where c_i is the unit cost of the i^{th} raw-material, c_{ij} is the unit transportation cost on the arc (i, j), and d_t is the minimum demand of the final product.

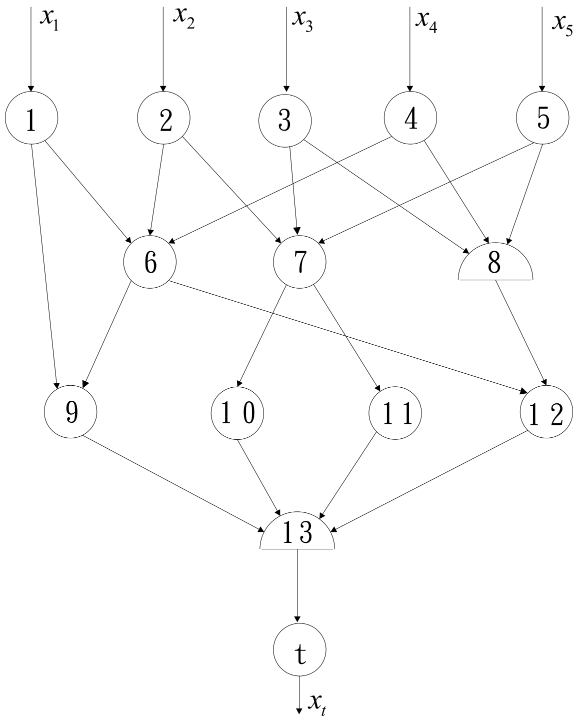

Figure 2.9: An Assembly Network

Fang and Qi (2003) propose a modified network simplex algorithm to solve the minimum distribution cost problem. In this thesis, we define three shortest path problems based on the minimum distribution cost problem and develop new algorithms to solve them.

2.3 Traditional Shortest Path Problems and Algorithms

The traditional shortest path problem usually seeks a path of minimum cost (or length) from a specific source node to termination nodes where each arc $(i, j) \in A$ is associated with a cost (or length) c_{ij}. The shortest path problem appears very often in many real-world applications, such as industrial networks, transportation networks, communication networks, and distribution networks etc. (Ahuja, Magnanti, and Orlin, 1993)

There are four types shortest path problems in the literature: (1) finding shortest paths from one source node to all other nodes when arc lengths are nonnegative; (2) finding shortest paths from one source node to all other nodes for networks with arbitrary arc lengths; (3) finding shortest paths from every node to every other node; and (4) various generalizations of the shortest path problem. We refer to problem types (1) and (2) as the single-source shortest path problems (SSSP). Depending on the number of termination nodes, SSSP seeks shortest paths either from one source node to one termination node (called one-to-one) or from one source node to all termination nodes (called one-to-some). The problem of type (3) is usually called as the all-pairs shortest path problem (APSP) or the all-to-all shortest path problem. The variations of the shortest path problem can be considered in many applications, for example, the maximum reliability path problem or shortest paths with time window constraints, etc. Our problem can also be viewed as shortest path problems with side constraints.

Most SSSP algorithms in the literature can be classified into two groups: *label setting algorithms* and *label correcting algorithms.* They are iterative procedures and can be used only on graphs without negative cycles. Label-setting algorithms are applicable for the shortest path problems with nonnegative arc lengths or the shortest path problems defined on acyclic networks with arbitrary arc lengths. Label-correcting algorithms are more general applicable for problems with arbitrary arc lengths.

The most popular label-setting algorithm is Dijkstra's algorithm (Dijkstra, 1959), as will be illustrated in Section 2.4. A naive implementation of Dijkstra's algorithm takes $O(n^2)$ time. Dial's implementation of Dijkstra's algorithm (Dial, 1969) gives better running time $O(m + nC)$ for graphs with small arc lengths. In particular, Dial's algorithm stores nodes in $nC + 1$ buckets, where C is the largest arc length (cost) in the network and nC is an upper bound on the distance label. Nodes with the same distance labels are stored in the same bucket whose index represents the distance label. The algorithm extracts nodes from a nonempty bucket, in the order of ascending index, scans outgoing arcs from these nodes, updates the distance label of the tail node for each outgoing arc, and then redistributes these nodes to new buckets. These procedures are repeated until all buckets become empty. To improve Dial's implementation, Ahuja et al. (Ahuja, Mehlhorn, Orlin, and Tarjan, 1990) use a radix heap data structure that shortens the time to check nonempty buckets and gives a running time of $O(m + n\log(nC))$.

There are several implementation of Dijkstra's algorithm based on the heap data structure, such as the binary-heap implementation (Johnson, 1972) with $O(m\log n)$ running time and Fibonacci heap implementation (Fredman and Tarjan, 1987) with $O(m + n\log n)$ running time. If we implement Dijkstra's algorithm using a heap, we need to create an empty heap to store nodes, find and return a node with minimum key in the heap, insert new nodes with predefined keys, reduce the key of a node from its current value to a new value, which must be smaller then the key it is replacing. and delete a node with minimum key from the heap.

Unlike label-setting algorithms, label-correcting algorithms can solve shortest path problems with negative arc lengths as long as there exists no negative cycle.

Ford (1956) outlined the first label-correcting algorithm for the shortest path problem. The FIFO implementation (Bellman, 1958) of the generic label-correcting algorithm stored nodes by using a queue data structure and takes $O(mn)$ time. The label-correcting algorithm is good for its flexibility. In particular, we can scan arcs in the network, one by one, and check the condition $d(j) > d(i) + c_{ij}$. Whenever an arc (i, j) satisfies the condition, we update $d(j) = d(i) + c_{ij}$. It stops when no distance label changes during an entire pass.

The Floyd-Warshall algorithm (Floyd, 1962) solves the all-pairs shortest path problem. It is a matrix-type algorithm and composed of three for loops with complexity $O(n^3)$. The algorithm uses $d^k(i, j)$ to represent the length of a shortest path from node i to node j that passes through the nodes 1,2,...,$k - 1$. It repeats the distance label update: $d^{k+1}(i, j) = min\{d^k(i, j), d^k(i, k) + d^k(k, j)\}$ for $k = 1, ..., n$ and all node pairs i and j.

This thesis considers distribution networks with nonnegative costs. Thus, we will develop algorithms based on Dijkstra's algorithm to solve uncapacitated minimum distribution cost problems which we view as specialized shortest path problems.

2.4 Dijkstra's Algorithm

Dijkstra's algorithm finds shortest paths from the source node s to all other nodes in a network with nonnegative arc lengths (or costs). Let $G = (N, A)$ be a network. The source node is s, S is the set of permanently labelled nodes, $\bar{S}$ is the set of temporarily labelled 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 to that node. Here we denote $d(i)$ to be the distance from node s to node i for each

$i \in N$, and pred(i) to be the predecessor of node i. Let $A(i)$ be the set of outgoing arcs of node i.

Algorithm *Dijkstra*;

```
Begin
    $S := \emptyset$; $\bar{S} := N$;
    $d(i) := \infty$ for each node $i \in N$;
    $d(s) := 0$ and pred$(s) := 0$;
    While $|S| < n$ do
        let $i \in \bar{S}$ be a node for which $d(i) = \min\{ d(j) : j \in \bar{S} \}$;
        $S := S \cup \{i\}$;
        $\bar{S} := \bar{S} - \{i\}$;
        for each $(i,j) \in A(i)$ do
            If $d(j) > d(i) + c_{ij}$ Then
                $d(j) := d(i) + c_{ij}$ and pred$(j) := i$;
    End While
End
```

Dijkstra's algorithm is a greedy algorithm, since each node selection operation selects a node i with the smallest distance label in $\bar{S}$, and then updates $d(j) = min\{d(j), d(i) + c_{ij}\}$ for each arc (i,j) in A.

2.5 Summary

We introduce the notations and the minimum distribution cost problem in a MNF model, and review the traditional shortest path problems and algorithms. The Dijkstra's algorithm is presented since we will develop algorithms for solving three uncapacitated minimum cost flow problems in a manufacturing network model with D-nodes based on the Dijkstra's algorithm.

CHAPTER III

UNCAPACITATED MINIMUM DISTRIBUTION COST PROBLEM (ONE-TO-ONE CASES)

In this Chapter, we introduce three uncapacitated minimum distribution cost problems (UMDCP) which send flow from one source node to one termination node, called one-to-one UMDCP$_1$, one-to-one UMDCP$_2$, and one-to-one UMDCP$_3$. Most notations appeared in this Chapter are listed in Section 2.1. First we introduce a network compaction procedure which can be served as a preprocessing process to reduce the original distribution network into an equivalent one of smaller size. Then we discuss each of the three one-to-one UMDCP problems, give their formulation and provide solution method.

3.1 Preprocessing

We may transform a distribution network into an equivalent one with smaller size by removing some nodes or arcs, or integrating some arcs. There are three possible cases for 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.

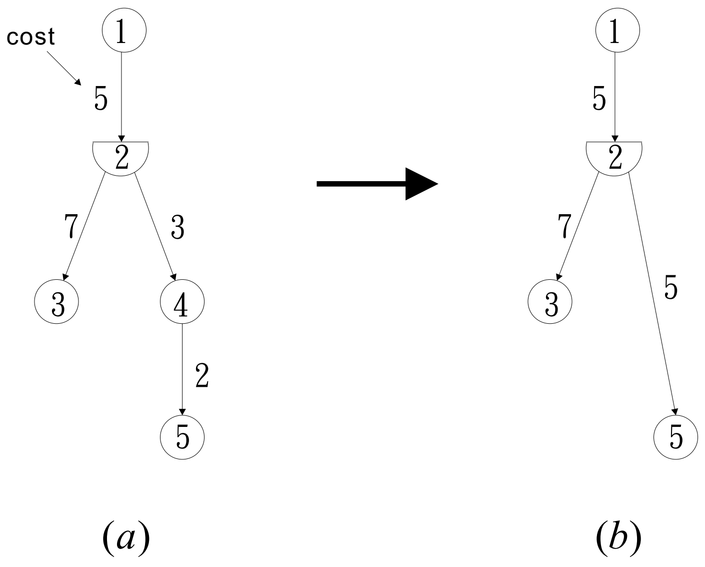

Figure 3.1: Any O-node with only one incoming arc and one outgoing arc can be compacted.

Taking the distribution network in Figure 3.1(a) for example. After removing node 4 and merging arc (2,4) and (4,5), we have arc (2,5) with cost $3 + 2 = 5$ as shown in Figure 3.1(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.

- **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$_2$ and UMDCP$_3$, which considers the distribution ratios of a D-node and arc flows.

Case 2.1: In UMDCP$_1$, 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 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 G'.

Step 1 Identify all the O-nodes (say, $v_1, \ldots, v_n$) adjacent to this D-group. For each arc (j_a, j_b) that connects D-nodes j_a and j_b in 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 (j_0, v_i), remove all the other arcs connecting to or within G'.

We take Figure 3.2(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 $\frac{c_{24}}{3}$ and c_{46} to be $\frac{c_{46}}{2}$ as shown in Figure 3.2(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 $(2,8)$ are $1+2=3$, $1+2+2=5$ and $1+2+5=8$.

Case 2.2: In our UMDCP$_2$ and UMDCP$_3$, 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:

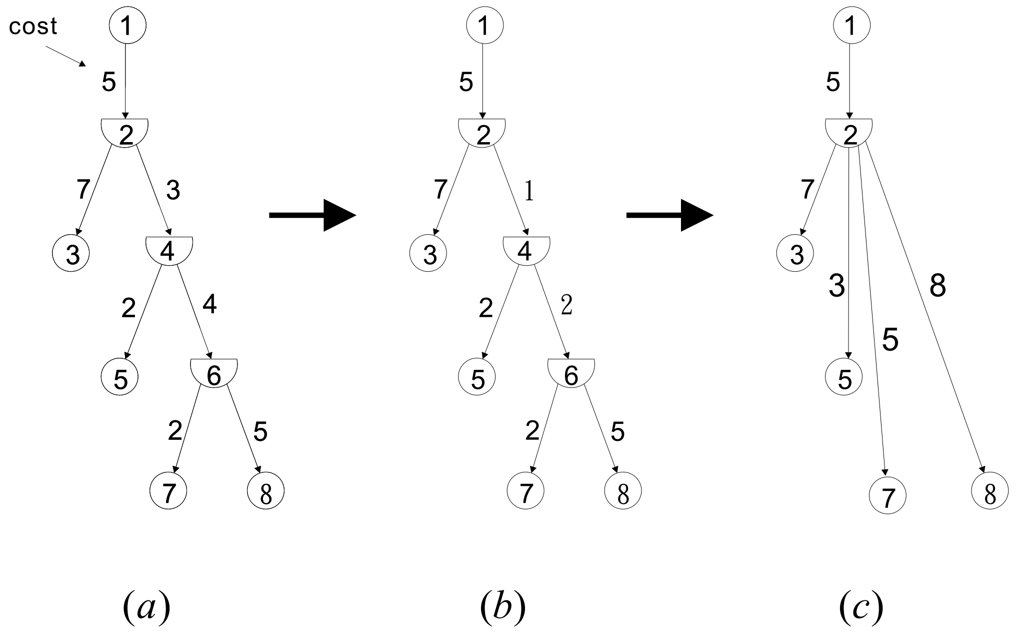

Figure 3.2: Adjacent D-nodes can be compacted into one D-node in UMDCP$_1$

Step 0 Identify a D-group 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 G'.

Step 1 Identify all the O-nodes (say, $v_1, \dots, v_n$) 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 $c_{j_0 v_i} = \sum_{(u,v) \text{ in the path } j_0 \to v_i} c_{uv}$ and distillation ratio $k_{j_0 v_i} = \prod_{(u,v) \text{ in the path } j_0 \to v_i} k_{uv}$.

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 G'.

For example, in Figure 3.3(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 $k_{25} = 0.8 \cdot 0.4 = 0.32$; $c_{27} = 3+4+2 = 9$ and $k_{27} = 0.8 \cdot 0.6 \cdot 0.3 = 0.144$; $c_{28} = 3+4+5 = 12$ and $k_{28} = 0.8 \cdot 0.6 \cdot 0.7 = 0.336$.

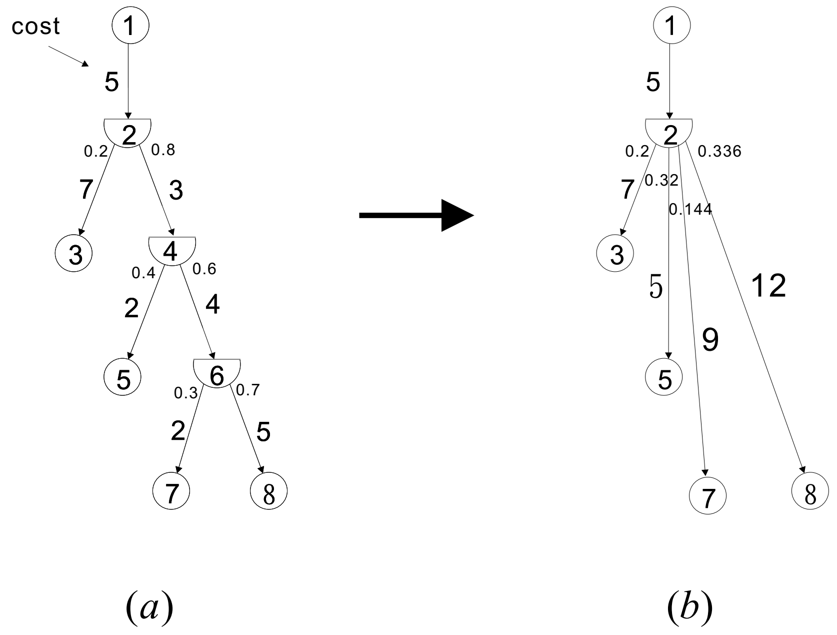

Figure 3.3: Adjacent D-nodes can be compacted into one D-node in UMDCP$_2$ and UMDCP$_3$

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

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 3.4, Figure 3.5 and Figure 3.6 for examples. One arc can be reduced each time in case 3, thus case 3 will be executed $O(m)$ times.

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 any D-group exists.

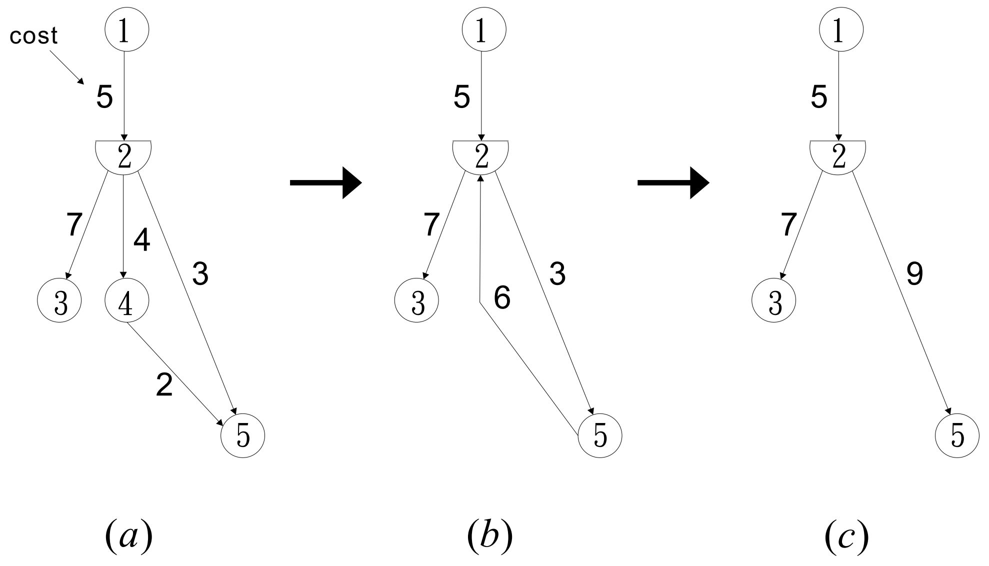

Figure 3.4: Parallel arcs connecting to the same nodes can be compacted (1)

3.2 One-to-one UMDCP$_1$

Our first uncapacitated minimum distribution cost problem, UMDCP$_1$, finds the minimum cost to connect a source node (S-node) to a termination node (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 distillation constraints.

For the one-to-one UMDCP$_1$, once a D-node is selected in a main path connecting the source s and termination 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. An example of one-to-one UMDCP$_1$ and its solution are shown in Figure 3.7(a) and (b).

3.2.1 Formulation and Solution Method for a One-to-one UMDCP$_1$

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 arc (i, j). When $(i, j)'$ appears in a solution to a one-to-one UMDCP$_1$, it means the D-node i is in the main path connecting s and

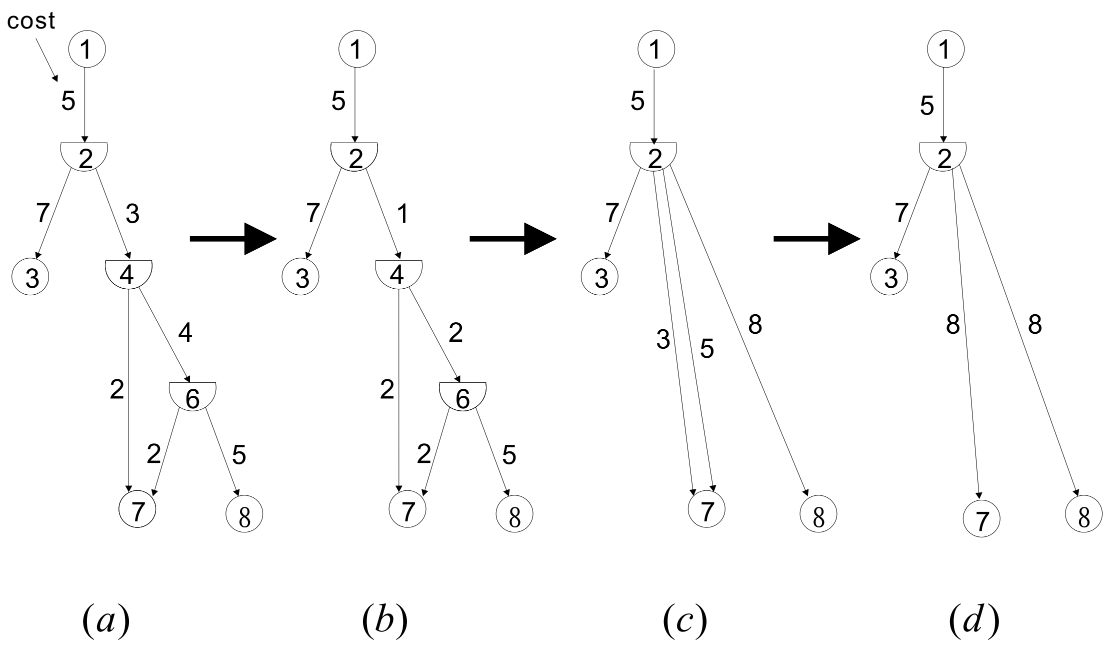

Figure 3.5: Parallel arcs connecting to the same nodes can be compacted (2)

t but the O-node j is not (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 (j can be viewed as a major product). So, 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 be in the main path and thus the fixed cost of (i, j) (or $(i, j)'$) has to be made. For each termination node i in N_T, if i is the requested termination node, we set its demand $d(i) = 1$, otherwise $d(i) = 0$. Let B be the set of all arcs $(i, j)'$ for $i \in N_D$ and $j \in L(i)$. We can formulate a UMDCP$_1$ problem as follows:

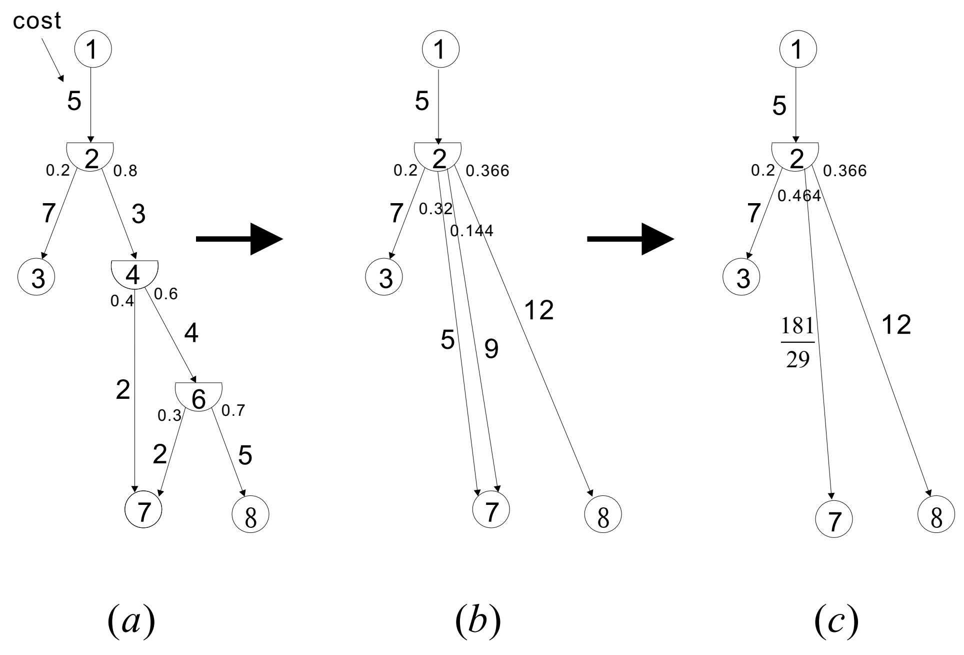

Figure 3.6: Parallel arcs connecting to the same nodes can be compacted (3)

$$
\begin{array}{ll}
\min & \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{(i,j)\in B} c_{ij}x'_{ij} \\
\textit{subject to} & \sum_{k\in L(i)} x_{ik} = \sum_{j\in E(i)} x_{ji} \quad \forall i \in N_O \\
& \sum_{j\in E(i)} x_{ji} = d(i) \quad \forall i \in N_T \\
& x_{ji} = x_{ik} + x'_{ik} \quad \forall j \in E(i), \quad \forall k \in L(i), \quad \forall i \in N_D \\
& \sum_{(i,k)\in A(i)} x_{ik} \le 1 \quad \forall i \in N_D \\
& \text{all } x_{ij} \text{ and } x^{'}_{ij} \text{ are binary}
\end{array}
\qquad (3.1)
$$

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 all O-nodes. The second constraint requires all T-nodes to satisfy their demands. The third and forth constraints are relative to D-nodes which show that only one of the original arc and the 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

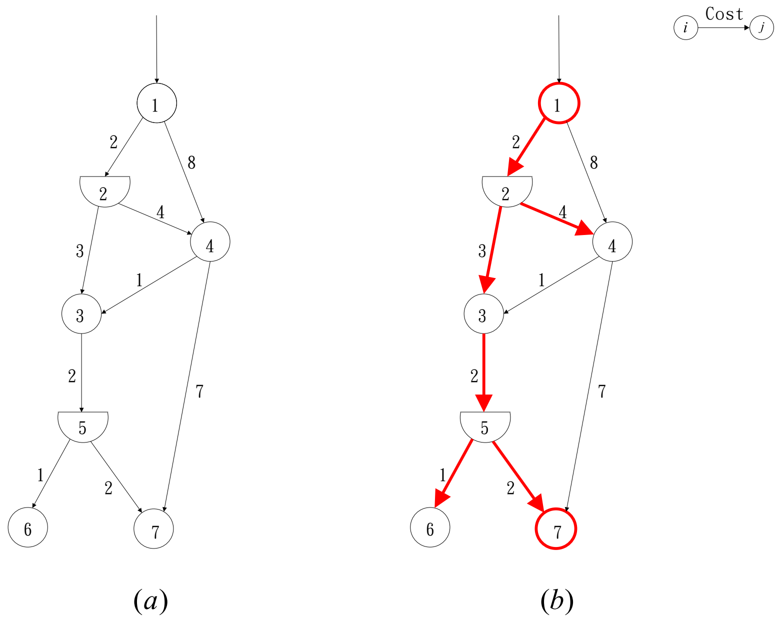

Figure 3.7: An UMDCP$_1$ problem

selected, if $x_{ij} = 1$ or $x'_{ij} = 1$; arc (i, j) is not selected if $x_{ij} = 0$ and $x'_{ij} = 0$.

This 0-1 integer programming problem of course can be solved by some optimization softwares like Lindo, Lingo or CPLEX. Here we present a combinatorial algorithm to solve it in next Section.

3.2.2 Algorithm for Solving a One-to-one UMDCP$_1$

Based on Dijkstra's Algorithm which solves the traditional shortest path problem, we develop a new algorithm to solve our one-to-one UMDCP$_1$ problem.

Let $G = (N, A)$ be a distribution network, N_O and N_D be the set of O-nodes and D-nodes. The source node is s. 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. Denote c_{ij} to be the length (or cost) of each arc

$(i,j) \in A$, $d(i)$ to be the distance from node s to node i for $i \in N$, and pred(i) to be the predecessor of node i. $A(i)$ is the set of outgoing arcs of node i.

Before running our algorithm, we observe that for any O-node i appeared in a solution where a main path connecting s and t passes i, 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, we only need to select one of its outgoing arcs. The proof is shown in Appendix A.

The algorithm to solve UMDCP$_1$ is briefly listed as below. Detailed steps are described in Appendix B.

Algorithm for solving a one-to-one UMDCP$_1$;

- ***Initialize***: All nodes are in $\bar{S}$ and $d(i)$ is infinity for each node i in N.
- ***step1***: If the source node s is an O-node, set $d(s) = 0$ and pred(s)=0.
- ***step2***: If the source node s is a D-node, add the lengths of all arcs which connect node s as its distance, and set pred(s)=0.
- ***step3***: Select node i with the minimum distance in $\bar{S}$, add node i to S and delete it from $\bar{S}$.
- ***step4***: If node i is a D-node, go to step3.
- ***step5***: If node i is an O-node but node j is a D-node for each outgoing arc (i,j) of node i, add $d(i)$, c_{ij}, and all lengths of the outgoing arcs (j,k) of node j as

$d(j)$, and set pred(j) is i. If $d(k)$ is larger than $d(j)$, update $d(k)$ to be $d(j)$, and set pred(k)=j.

- ***step6***: If node i and node j are both O-nodes for each outgoing arc (i, j) of node i and if $d(j)$ is larger than the summation of $d(i)$ and $c_{i,j}$, then update $d(j)$ to be $d(i) + c_{ij}$, and set pred(j)=i.

- ***step7***: Repeat *step3* to *step6* until all nodes are removed to S.

The complexity of our one-to-one UMDCP$_1$ algorithm is $O(n^2)$. In particular, our one-to-one UMDCP$_1$ algorithm performs the following two basic operations: (1) *Node selections*: one-to-one UMDCP$_1$ algorithm performs this operation n times and each node-selection operation requires to scan each node in the temporary set. Hence, the total time of node-selection is $n + (n-1) + (n+2) + ... + 1 = O(n^2)$. (2) *Distance updates*: UMDCP$_1$ algorithm performs this operation $|A(i)|$ times for each node i. Overall, the algorithm performs this operation $\sum_{i \in N} \mid A(i)| = m$ times, and each distance-update operation requires $O(1)$ time. Thus, the total time for updating all distance labels is $O(m)$.

We have the total time $O(n^2 + m) = O(n^2)$ for implementing the one-to-one UMDCP$_1$ algorithm. Actually, the running time of the algorithm depends on how the min-priority queue is implemented. In general, the running time is $S(n, m, C)$, a function of n, m and $C = \max_{(i,j) \in A} \{|C_{ij}|\}$.

3.2.3 Correctness of a One-to-one UMDCP$_1$ Algorithm

Let $G = (N, A)$ be a general distribution network with a source node s and a specific termination node u, and S be the set of permanently labeled nodes, $S \cup \overline{S} = N$. Denote the distance from s to u as $d(u)$ and the distance of the shortest path as $\delta(s, u)$.

We introduce some properties. The process of relaxing an arc (u, v) consists of testing whether we can improve the shortest path to v found so far by going through u, and if so, updating $d(v)$ and pred(v).

Upper-bound property:

We always have $d(v) \geq \delta(s, v)$ for all nodes $v \in N$ and once $d(v)$ achieves the value $\delta(s, v)$, it never changes.

No-path property:

If there is no path from s to v, then we always have $d(v) = \delta(s, v) = \infty$.

Convergence property:

If $s \hookrightarrow u \rightarrow v$ is a shortest path in G for some $u, v \in N$, and if $d(u) = \delta(s, u)$ at any time prior to relaxing arc (u, v), then $d(v) = \delta(s, v)$ at all times afterward.

After running our modified Dijkstra's algorithm, we assert that $d(u) = \delta(s, u)$ for all u. Note that once u is added to S, $d(u)$ is not changed and should be $\delta(s, u)$. We rely on the upper-bound property to show that $d(u) = \delta(s, u)$ holds at all times.

Now we prove the correctness of our one-to-one UMDCP$_1$ algorithm by contradiction. Suppose that u is the first node added to S for which $d(u) \neq \delta(s, u)$. Because s is the first node added to S and $d(s) = \delta(s, s) = 0$ at that time, we must have $u \neq s$. We also have $S \neq \emptyset$ before u is added to S because $u \neq s$. There must have some paths from s to u, because we have $d(u) = \delta(s, u) = \infty$ by the no-path property which violates our assumption that $d(u) \neq \delta(s, u)$. Since there is at least one path, it must have a shortest path from s to u.

We consider that p is a shortest path from s to u. If all nodes along path p are O-nodes, the proof is similar to the proof of correctness for the traditional Dijkstra's algorithm (Cormen, Leiserson, Rivest, and Stein, 2001). Otherwise, we have two situations to discuss. In Figure 3.8(a): Let x be a D-node in S, y be an O-node and the first node along p not within S, and z also be an O-node not in S. Thus, p can be decomposed as $s \to (p_1) \to x \to (x, y) \to y \quad (x \to (x, z) \to z) \quad \to (p_2) \to u$ because x is a D-node, and x is the predecessor of y and z. In Figure 3.8(b): Let x be an O-node in S, w be an D-node and the first node along p not within S, y and z be O-nodes not in S next to w. Hence, the predecessors of w, y, z are x, w and w, and p can be decomposed as $s \to (p_1) \to x \to (x, w) \to w \to (w, y) \to y \quad (w \to (w, z) \to z)$ $\to (p_2) \to u$.

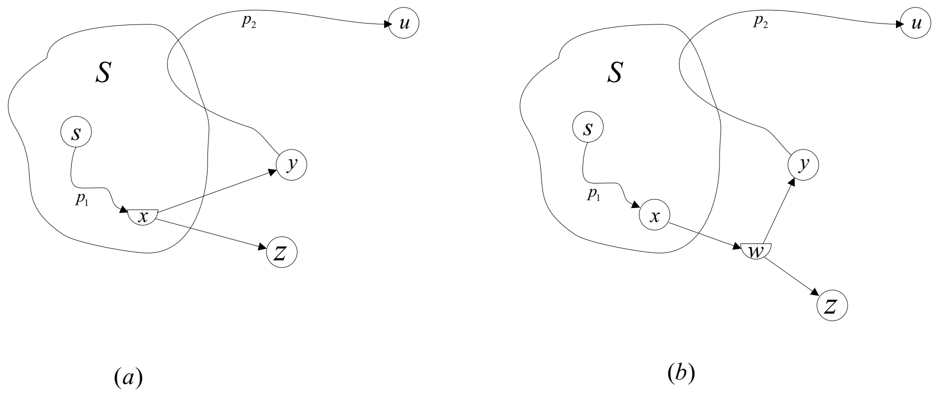

Figure 3.8: The proof of our UMDCP$_1$ algorithm.

In Figure 3.8(a), when x is added to S, $d(x) = \delta(s, x)$. By the Convergence property, arc (x, y) is relaxed, thus $d(y) = \delta(s, y)$ so that $d(y) = \delta(s, y) \leq \delta(s, u) \leq d(u)$. And, $d(x) = d(y) = d(z)$ because x is a D-node. However, since both y and u are in $\overline{S}$, we will have $d(u) \leq d(y)$ when u is selected. Thus, the two inequalities must be equalities, $d(y) = \delta(s, y) = \delta(s, u) = d(u)$. Clearly, $d(u) = \delta(s, u)$ contradicts our assumption. We can conclude that each u is added to S, $d(u) = \delta(s, u)$.

Then, in Figure 3.8(b), it is the same as the case when x is added to S, $d(x) = \delta(s, x)$. By the Convergence property again, arc (x, w) is relaxed, then $d(w) = \delta(s, w)$ so that $d(w) = \delta(s, w) \leq \delta(s, u) \leq d(u)$. Here, because w is a D-node, when we calculate $d(w)$, we must add $d(w, y)$ and $d(w, z)$ to it and have $d(w) = d(y) = d(z)$. Similarly, both w and u are in $\overline{S}$, we have $d(u) \leq d(w)$ when u is selected. So, we have the equality $d(w) = \delta(s, w) = \delta(s, u) = d(u)$. And, $d(u) = \delta(s, u)$ still contradicts our assumption. Thus, $d(u) = \delta(s, u)$ when u is added to S.

In a word, we choose a node i in $\bar{S}$ with the smallest $d(i)$ to calculate in each step of our one-to-one UMDCP$_1$ algorithm. Hence, following the step until to the specific termination node u is permanently labeled, we can obtain the smallest $d(u)$ as its $\delta(s, u)$. That is, the path is the shortest path from s to u, and the minimum total cost is $\delta(s, u)$.

3.2.4 A One-to-one UMDCP$_1$ Example

We take Figure 3.7(a) as our example, show its formulation and optimal solution by software. Then we use our algorithm to calculate the minimum total cost and find the shortest paths.

In Figure 3.7(a), we want to find the shortest path from node 1 to node 7. The formulation of this example can be shown as follow:

$$\begin{array}{ll}
\min & 2x_{12} + 8x_{14} + 3x_{23} + 3x'_{23} + 4x_{24} + 4x'_{24} + 2x_{35} \\
& +x_{43} + 7x_{47} + x_{56} + x'_{56} + 2x_{57} + 2x'_{57} \\
\textit{subject to} & x_{12} + x_{14} = 1 \\
& x_{23} + x_{43} = x_{35} \\
& x_{14} + x_{24} = x_{43} + x_{47} \\
& x_{56} = 0 \\
& x_{57} + x_{47} = 1 \\
& x_{12} = x_{23} + x'_{23} \ , \quad x_{12} = x_{24} + x'_{24} \ , \quad x_{23} + x_{24} \leq 1 \\
& x_{35} = x_{56} + x'_{56} \ , \quad x_{35} = x_{57} + x'_{57} \ , \quad x_{56} + x_{57} \leq 1 \\
& \text{all } x_{ij} \text{ and } x'_{ij} \text{ are binary}
\end{array}$$

Using CPLEX, we obtain an optimal solution: $x^*_{12} = 1$, $x^*_{23} = 1$, $x'^*_{24} = 1$, $x^*_{35} = 1$, $x'^*_{56} = 1$, $x^*_{57} = 1$, and 0 for other variables. Thus a shortest path contains arcs (1,2), (2,3), (2,4), (3,5), (5,6), (5,7). Another optimal solution is $x^*_{14} = 1$, $x^*_{43} = 1$, $x^*_{35} = 1$, $x'^*_{56} = 1$, $x^*_{57} = 1$, represented by a shortest path: (1,4), (4,3), (3,5), (5,6), (5,7). The minimum total cost for these two paths are both 14.

Next, we solve the same problem by our algorithm. Initially, $d(1)$=0, pred(1)=0 and $d(i) = \infty$ for each node i other than node 1. Starting from node 1, suppose we pass arc (1,2) first. Since node 2 is a D-node, we add up the costs of arc (1,2), (2,3) and (2,4), and have $d(2)$=2+3+4=9, $d(3)$=$d(4)$=9. Since 9 is smaller than infinity, we update node 2, 3, 4, and set pred(2)=1, pred(3)=pred(4)=2. Then, suppose we pass arc (1,4) and calculate $d(4)$=8, which is smaller than its previous distance label 9. Thus, we update node 4 to have $d(4)$=8 and pred(4)=1.

Following the same steps to update the distance value and predecessor for all nodes, we can obtain $d(7)$=14 and trace the predecessors $7 \rightarrow 5 \rightarrow 3 \rightarrow 2 \rightarrow 1$ to obtain a path (1,2), (2,3), (3,5), (5,7). Since node 2 and node 5 are D-nodes, arc (2,4) and (5,6) must also be included. Thus, the shortest path for this example is (1,2), (2,3), (2,4), (3,5), (5,6), (5,7) and the minimum total cost from node 1 to node 7 is 14. Note that we have another solution: (1,4), (4,3), (3,5), (5,6), (5,7) with total cost 14 as shown in Figure 3.9.

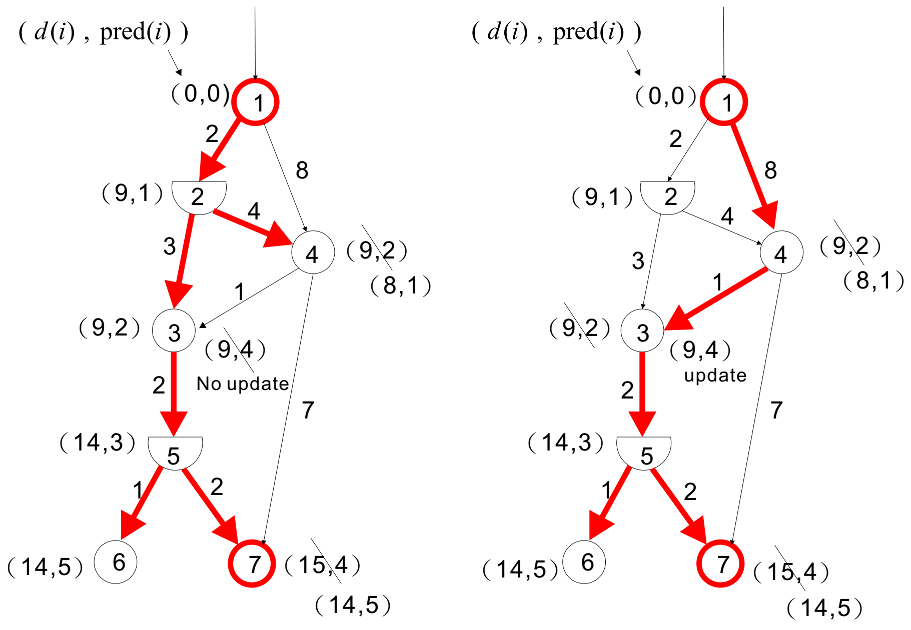

Figure 3.9: The shortest path solved by the UMDCP$_1$ algorithm

3.2.5 A Reduced Method for a One-to-one UMDCP$_1$

Here, we introduce a reduced method to solve our one-to-one UMDCP$_1$. 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 length to the incoming arc of the D-node, and reset the length of all the outgoing arcs of the D-node to be 0. Then, all D-nodes can be transformed to be O-nodes, and we can regard the new network as a traditional network without D-nodes, which can be solved by

Dijkstra's algorithm. The transformed network is shown in Figure 3.10.

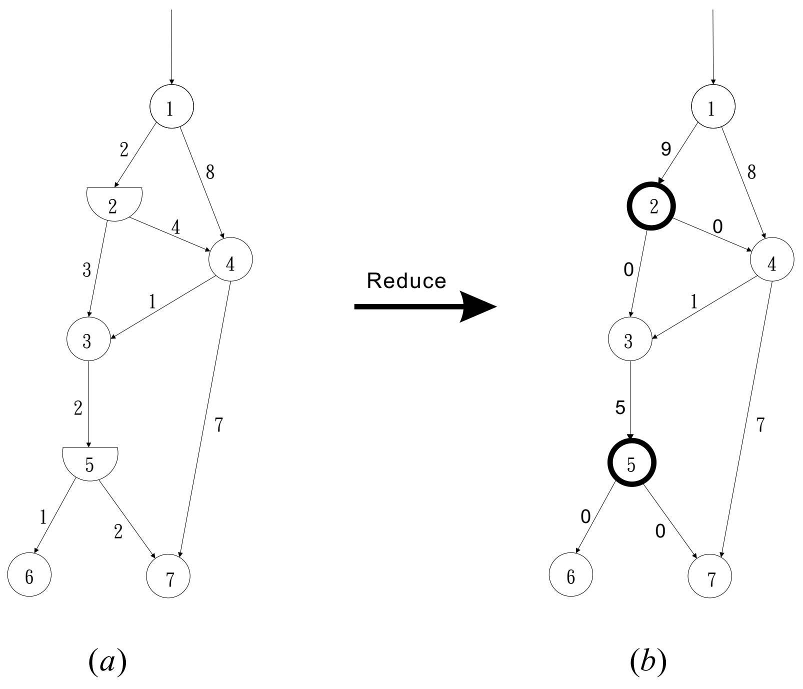

Figure 3.10: The original and reduced UMDCP$_1$ problems

The formulation of the one-to-one UMDCP$_1$ is shown in equation (3.2). All D-nodes are transformed to O-nodes, so we only need to consider the flow balance constraints.

Suppose a D-node has n outgoing arcs. By the third constraint, we have $x_{ik_1} + x_{ik_2} + ... + x_{ik_n} \leq 1$. 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 $p = 2, ..., n$. Thus, we know that $x_{ik_1} = x'_{ik_2} = ... = x'_{ik_n} = 1$ and $x'_{ik_1} = x_{ik_2} = ... = x_{ik_n} = 0$. By the second constraint $x_{ji} = x_{ik} + x'_{ik}$ for all k, we obtain $x_{ji} = x_{ik_1} + x'_{ik_1}$, $x_{ji} = x_{ik_2} + x'_{ik_2}$, ... , $x_{ji} = x_{ik_n} + x'_{ik_n}$. Sum up the n equations, we have $nx_{ji} = x_{ik_1} + x_{ik_2} + ... + x_{ik_n} + x'_{ik_1} + x'_{ik_2} + ... + x'_{ik_n}$. Since $x_{ik_1} = x'_{ik_2} = ... = x'_{ik_n}$ and $x'_{ik_1} = x_{ik_2} = ... = x_{ik_n} = 0$, we have $nx_{ji} = nx_{ik_1}$. Hence, $x_{ji} = x_{ik_1}$, i.e., $x_{ji} = x_{ik_1} + x_{ik_2} + ... + x_{ik_n} = \sum_{(i,k)\in A(i)} x_{ik}$. This satisfies the flow balance constraint for an O-node.

So, we can transform the one-to-one UMDCP$_1$ as a traditional one-to-one shortest path problem.

$$
\begin{array}{ll}
\min & \sum_{(i,j)\in A} c_{ij}x_{ij} \\
\textit{subject to} & \sum_{k\in L(i)} x_{ik} = \sum_{j\in E(i)} x_{ji} \quad \forall i \in N_O \\
& \sum_{j\in E(i)} x_{ji} = d(i) \quad \forall i \in N_T \\
& \text{all } x_{ij} \text{ are binary}
\end{array}
\qquad (3.2)
$$

In the transformed network, the main difference is that we transform all D-nodes to O-nodes. Take Figure 3.11 for example, we sum up the costs c_{ji}, c_{ik_1}, ... , c_{ik_n} on all arcs connecting to a D-node i as the cost of the incoming arc (j,i) of node i, and reset the cost of each outgoing arc (i,k) of node i to be 0. Intuitively speaking, when we select a D-node, we also have to select all its outgoing arcs. That is, we can regard the total cost from node j to D-node i, and i to k_1, ... , and k_n as a new cost from node j to node i, and set all other outgoing arcs from D-node i to have zero cost.

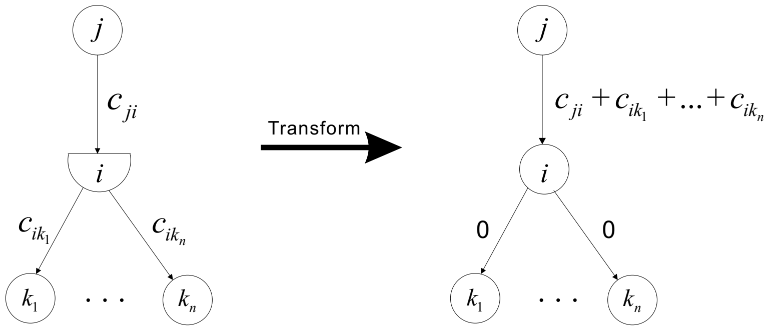

Figure 3.11: Transform a D-node to an O-node

Take the network in Figure 3.7(a) for example. Its reduced network is shown in Figure 3.10 with formulation as following:

$$\begin{aligned}
\min \quad & 9x_{12} + 8x_{14} + x_{43} + 5x_{35} + 7x_{47} \\
\textit{subject to} \quad & x_{12} + x_{14} = 1 \\
& x_{12} = x_{23} + x_{24} \\
& x_{23} + x_{43} = x_{35} \\
& x_{14} + x_{24} = x_{43} + x_{47} \\
& x_{35} = x_{56} + x_{57} \\
& x_{56} = 0 \\
& x_{47} + x_{57} = 1 \\
& \text{all } x_{ij} \text{ are binary}
\end{aligned}$$

Solving this integer program (in fact, if we remove the binary integer constraint, the relaxed LP will have the same solution) by CPLEX, the solution is $x^*_{12} = 1$, $x^*_{23} = 1$, $x^*_{35} = 1$, $x^*_{57} = 1$, and others are 0. So, the main shortest path contains arcs (1,2), (2,3), (3,5), (5,7). Because node 2 and 5 are D-nodes, arcs (2,4) and (5,6) must also be selected. Hence, the optimal solution contains arcs (1,2), (2,3), (2,4), (3,5), (3,6), and (5,7), with the optimal objective value 14. Another solution is (1,4), (4,3), (3,5), (5,6), (5,7) with objective value 14, too.

Next, we use Dijkstra's algorithm to solve this example. Tracing the predecessor $7 \to 5 \to 3 \to 2 \to 1$, we also have a main shortest path (1,2), (2,3), (3,5), (5,7). And, because the main shortest path passes D-nodes 2 and 5, the path must also include arcs (2,4) and (5,6). Thus the traditional Dijkstra's algorithm give the same optimal solution as the one by CPLEX.

3.2.6 Summary

Section 3.2 describes our one-to-one UMDCP$_1$ problem and gives its formulation that focus on a main path connecting the origin and destination node. This problem considers the dependency relationship of arcs and ignores the flow distribution ratios associated with D-nodes. We use CPLEX to calculate an optimal one-to-one UMDCP$_1$ solution, and also develop a modified Dijkstra's algorithm which also obtains an optimal solution. Then we propose a reduced method that transforms the network and formulation to a traditional one-to-one shortest path problem which is solvable by traditional Dijkstra's algorithm.

3.3 One-to-one UMDCP$_2$

Our second uncapacitated minimum distribution cost problem, UMDCP$_2$, finds the minimum cost to receive a unit flow for a requested T-node from an S-node. Unlike UMDCP$_1$ which only consider 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.

For the one-to-one UMDCP$_2$, once a D-node appears in a main path connecting the source s and termination 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. An example of UMDCP$_2$ and its solution are shown in Figure 3.12(a) and (b).

3.3.1 Formulation and Solution Method for a One-to-one UMDCP$_2$

To formulate 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. When a D-node i lies on a main path connecting the requested source and termination nodes, only one of its outgoing arcs will appear in the main path. All the other outgoing arcs of i will receive flows according to the distribution ratios, but these flows will then be shipped to a dummy termination. In that case, we say the flow passes (i, j) for the outgoing arc in the main path but the flow passes $(i, j)'$ for the other

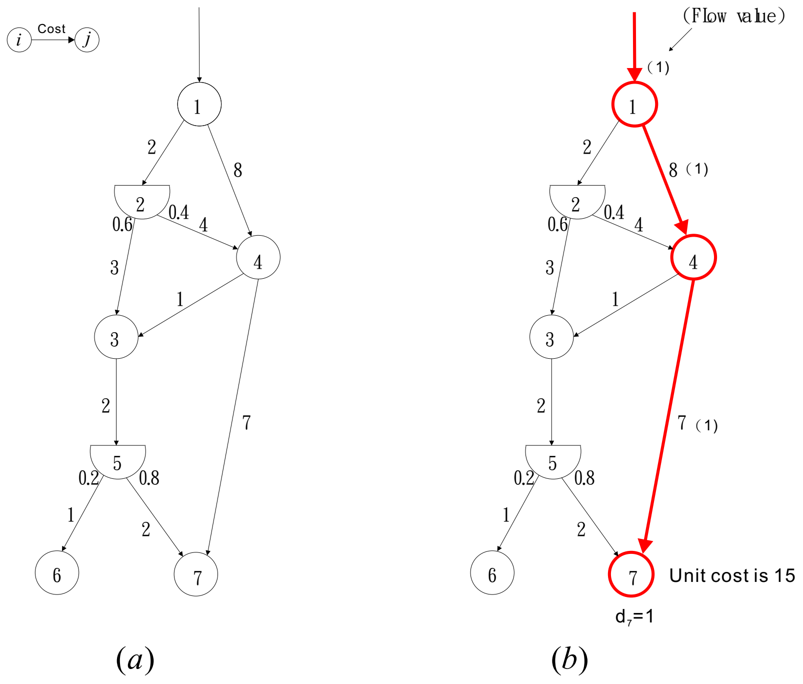

Figure 3.12: An UMDCP$_2$ problem

outgoing arcs of i. For each termination node i we associate it with a demand $d(i)$. For each termination node i in N_T, if i is the requested termination node, $d(i) \geq 1$, otherwise $d(i) \geq 0$. 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:

$$
\begin{array}{ll}
\min & \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{(i,j)\in B} c_{ij}x'_{ij} \\
\textit{subject to} & \sum_{k\in L(i)} x_{ik} = \sum_{j\in E(i)} x_{ji} \quad \forall i \in N_O \\
& \sum_{(j,i)\in E(i)} x_{ji} \geq d_i \quad \forall i \in N_T \\
& r_{ik}x_{ji} = x_{ik} + x'_{ik} \quad \forall j \in E(i) \ \ \forall k \in L(i) \ \ \forall i \in N_D \\
& x_{ji} - My_{ji} \leq 0 \quad \forall j \in E(i) \ \ \forall i \in N_D \\
& x_{ik} - My_{ik} \leq 0 \quad \forall k \in L(i) \ \ \forall i \in N_D \\
& x'_{ik} - My'_{ik} \leq 0 \quad \forall k \in L(i) \ \ \forall i \in N_D \\
& y_{ji} = y_{ik} + y'_{ik} \quad \forall j \in E(i), \ \ \forall k \in L(i) \ \ \forall i \in N_D \\
& \sum_{(i,k)\in L(i)} y_{ik} \leq 1 \quad \forall i \in N_D \\
& \text{all } y_{ij} \text{ and } y'_{ij} \text{ are binary}
\end{array}
\tag{3.3}
$$

The objective function finds the minimum total cost for all arcs in G', including those parallel arcs in the network. The first constraint is the flow balance constraint for all O-nodes. The second one shows that all T-nodes must satisfy their demand. The third constraint represents the flows on D-nodes' outgoing arcs should obey the distribution ratios. The forth, fifth and sixth constraints 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 $x_{ij} = 0$. The seventh and eighth ones 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 that arc (i,j) is selected, otherwise $y_{ij} = y'_{ij} = 0$ means arc (i,j) is not selected.

3.3.2 Algorithm for Solving a One-to-one UMDCP$_2$

Besides the notations in Section 3.2.2, each arc $(i,j) \in A$ and each node $i \in N$ has its own flow value, denoted by x_{ij} and x_i, respectively. 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 to solve a one-to-one UMDCP$_2$ as follows: (Detailed procedures are listed in Appendix C.)

Algorithm for solving a one-to-one UMDCP$_2$;

- ***Initialize***: All nodes are in $\bar{S}$, $d(i)$ and $avg\_d(i)$ are both infinity for each node i in N.
- ***step1***: If the source node s is an O-node, then set $d(s) = 0$, $avg\_d(i) = 0$, and pred(s)=0. Node s has an initial supply x_s, and we flood each outgoing arc (s, j) and each node j of node s by x_s.
- ***step2***: If the source node s is a D-node, node s has an initial supply x_s, and we flood each of its outgoing arc (s, j) by multiplication with its distribution ratio r_{sj} so that node j receives $r_{sj}x_s$ amount of flow. Then we have $d(s) = \sum_{(s,j)\in A(s)} r_{sj}x_s$, and each $d(j)$ equals $d(s)$. Calculate $avg\_d(s) = \frac{d(s)}{x_s}$ and each $avg\_d(j) = \frac{d(j)}{x_j}$, then set pred($s$)=0 and pred($j$)=$s$.
- ***step3***: Select node i with the minimum average distance in $\bar{S}$, add node i to S and delete it from $\bar{S}$.
- ***step4***: If node i is a D-node, then go to step3.
- ***step5***: If node i is an O-node connecting to a D-node j, the flow value x_{jk} of each outgoing arc (j, k) of node j can be calculated by $r_{jk}x_{ij}$. Let x_j and x_k equal to x_{ij} and x_{jk}, then $d(j) = d(i) + c_{ij} + \sum_{(j,k)\in A(j)} r_{jk}c_{jk}$. Calculate $avg\_d(j) = \frac{d(j)}{x_j}$ and set pred(j)=i. If the original $avg\_d(k)$ is larger than $avg\_d(j)$, update $d(k) = d(j)$, $avg\_d(k) = avg\_d(j)$ and set pred(k)=j.

- ***step6***: If node i is an O-node connecting to an O-node j, let x_{jk} for each outgoing arc (j,k) of node j equal x_{ij}, and $x_j = x_{ij}$, $x_k = x_{jk}$. And if the original $avg\_d(j)$ is larger, then update $d(j) = d(i) + c_{ij}$, $avg\_d(j) = \frac{d(j)}{x_j}$, and set pred($j$)=$i$.

- ***step7***: Repeat *step3* to *step6* until all nodes are removed to S.

The complexity of our one-to-one UMDCP$_2$ algorithm is $O(n^2)$ since it also performs the same procedures as the one-to-one UMDCP$_1$ algorithm in Section 3.2.2.

3.3.3 Correctness of our One-to-one UMDCP$_2$ Algorithm

Let $G = (N, A)$ be a general distribution network with a source node s and a specific termination node u, and S be the set of permanently labeled nodes, $S \cup \overline{S} = N$. Denote the distance from s to u as $d(u)$ and the length of the shortest path as $\delta(s,u)$. x_u is the flow value on node u. $avg\_d(u) = \frac{d(u)}{x_u}$ represents the average distance from s to u.

The three properties introduced in Section 3.2 can be used to prove our UMDCP$_2$ algorithm. After running our UMDCP$_2$ algorithm, we assert that $avg\_d(u) = avg\_\delta(s,u)$ for all u. Note that once u is added to S, $avg\_d(u)$ is not changed and should be $avg\_\delta(s,u)$. We rely on the upper-bound property to show that $avg\_d(u) = avg\_\delta(s,u)$ holds at all times.

Now we will prove the correctness of our UMDCP$_2$ algorithm by contradiction, and the proof is familiar to UMDCP$_1$. We also discuss three situations as listed in Figure 3.13. The main difference between UMDCP$_1$ and UMDCP$_2$ is that we need to consider the average distance in UMDCP$_2$.

In Figure 3.13(a), when x is added to S, $avg\_d(x) = avg\_\delta(s,x)$. By the Convergence property, arc (x,y) is relaxed at that time, then $d(y) = \delta(s,y)$ so that $avg\_d(y) = avg\_\delta(s,y) \leq avg\_\delta(s,u) \leq avg\_d(u)$. However, both y and u are in $\overline{S}$,

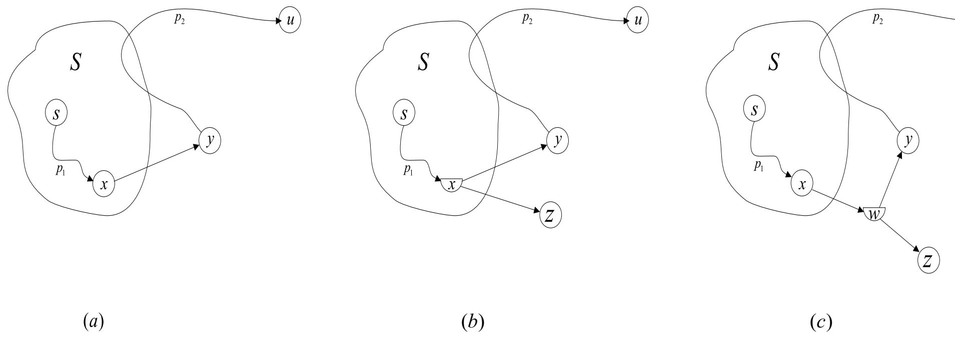

Figure 3.13: The proof of our UMDCP$_2$ algorithm.

when u is selected, we have $avg\_d(u) \leq avg\_d(y)$. Thus, the two inequalities must be equalities, $avg\_d(y) = avg\_\delta(s, y) = avg\_\delta(s, u) = avg\_d(u)$. Clearly, $avg\_d(u) = avg\_\delta(s, u)$ contradicts our assumption. We can conclude that each u is added to S, $avg\_d(u) = avg\_\delta(s, u)$.

Then, in Figure 3.13(b), when x is added to S, $avg\_d(x) = avg\_\delta(s, x)$. By the Convergence property, arc (x, y) is relaxed at that time, then $d(y) = \delta(s, y)$ so that $avg\_d(y) = avg\_\delta(s, y) \leq avg\_\delta(s, u) \leq avg\_d(u)$. And, $d(x) = d(y) = d(z)$ at that time because x is a D-node. However, both y and u are in $\overline{S}$, when u is selected, we have $avg\_d(u) \leq avg\_d(y)$. Thus, the two inequalities must be equalities, $avg\_d(y) = avg\_\delta(s, y) = avg\_\delta(s, u) = avg\_d(u)$. Clearly, $avg\_d(u) = avg\_\delta(s, u)$ contradicts our assumption. We can conclude that each u is added to S, $avg\_d(u) = avg\_\delta(s, u)$.

Last, in Figure 3.13(c), it is the same as the case when x is added to S, $avg\_d(x) = avg\_\delta(s, x)$. By the Convergence property again, arc (x, w) is relaxed at that time, then $d(w) = \delta(s, w)$ so that $avg\_d(w) = avg\_\delta(s, w) \leq avg\_\delta(s, u) \leq avg\_d(u)$. Here, because w is a D-node, when we calculate $d(w)$, we must add $d(w, y)$ and $d(w, z)$ to it and have $d(w) = d(y) = d(z)$. Similarly, both w and u are in $\overline{S}$, when u is selected, we have $avg\_d(u) \leq avg\_d(w)$. So, we have the equality $avg\_d(w) = avg\_\delta(s, w) = avg\_\delta(s, u) = avg\_d(u)$. And, $avg\_d(u) = avg\_\delta(s, u)$ still contradicts our assumption.

Thus, $avg\_d(u) = avg\_\delta(s, u)$ when u is added to S.

In a word, we choose the smallest $avg\_d(i)$ for each node i in $\bar{S}$ in our UMDCP$_2$ algorithm. Hence, following the step until to the specific termination node u, we can obtain the smallest $avg\_d(u)$ as its $avg\_\delta(s, u)$. That is, the path is the shortest path from s to u, and the minimum unit cost is $avg\_\delta(s, u)$, the minimum total cost equals to $avg\_\delta(s, u)$ multiplies the flow value on node u, x_u.

3.3.4 A One-to-one UMDCP$_2$ Example

We take Figure 3.12(a) as our example, show its formulation and optimal solution by software. Then we use our algorithm to calculate the minimum total cost for the termination to receive one unit of flow and find the shortest paths.

In Figure 3.12(a), we want to find the shortest path from node 1 to node 7. Let y be the supply in node 1. The mixed integer linear programming (MIP) formulation is as follows:

$$\begin{array}{ll}
\min & 2x_{12} + 8x_{14} + 3x_{23} + 3x'_{23} + 4x_{24} + 4x'_{24} + 2x_{35} \\
& +x_{43} + 7x_{47} + x_{56} + x'_{56} + 2x_{57} + 2x'_{57} \\
\textit{subject to} & x_{12} + x_{14} = y \\
& x_{23} + x_{43} = x_{35} \\
& x_{14} + x_{24} = x_{43} + x_{47} \\
& x_{56} \geq 0 \\
& x_{57} + x_{47} \geq 1 \\
& 0.6x_{12} = x_{23} + x'_{23} \ , \quad 0.4x_{12} = x_{24} + x'_{24} \\
& 0.2x_{35} = x_{56} + x'_{56} \ , \quad 0.8x_{35} = x_{57} + x'_{57} \\
& x_{12} - My_{12} \leq 0 \ , \quad x_{35} - My_{35} \leq 0 \\
& x_{23} - My_{23} \leq 0 \ , \quad x'_{23} - My'_{23} \leq 0 \\
& x_{24} - My_{24} \leq 0 \ , \quad x'_{24} - My'_{24} \leq 0 \\
& x_{56} - My_{56} \leq 0 \ , \quad x'_{56} - My'_{56} \leq 0 \\
& x_{57} - My_{57} \leq 0 \ , \quad x'_{57} - My'_{57} \leq 0 \\
& y_{12} = y_{23} + y'_{23} \ , \quad y_{12} = y_{24} + y'_{24} \ , \quad y_{23} + y_{24} \leq 1 \\
& y_{35} = y_{56} + y'_{56} \ , \quad y_{35} = y_{57} + y'_{57} \ , \quad y_{56} + y_{57} \leq 1 \\
& \text{all } y_{ij} \text{ and } y'_{ij} \text{ are binary}
\end{array}$$

Using CPLEX, we obtain an optimal solution: $y^* = 1$, $x^*_{14} = 1$, $x^*_{47} = 1$, and 0 for other variables. Hence, the shortest path is (1,4), (4,7) with unit flow, and the minimum total unit cost from node 1 to node 7 is 15.

We can solve the same example by our one-to-one UMDCP$_2$ algorithm. Initially, all $d(i) = \infty$ and $avg\_d(i) = \infty$. First step calculates $d(1) = 0$, $avg\_d(1) = 0$, and pred(1)=0 since node 1 is a source node. Node 1 has two outgoing arcs (1,2) and (1,4). Suppose we go through arc (1,2) first. Because node 2 is a D-node, we add

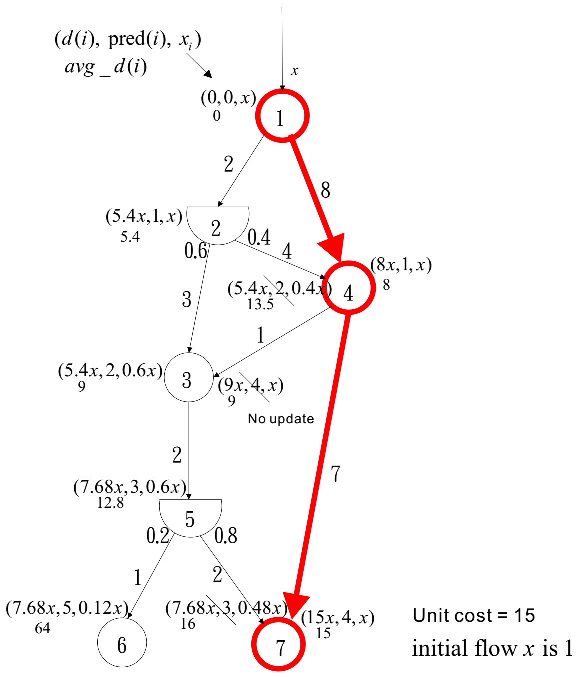

Figure 3.14: The shortest path solved by the UMDCP$_2$ algorithm

up the costs of arc (1,2), (2,3) and (2,4) by $d(2) = 2 \cdot y + 3 \cdot 0.6y + 4 \cdot 0.4y = 5.4y$, and set $d(3) = d(4) = 5.4y$. The flows enter node 2, 3, 4 are y, $0.6y$, and $0.4y$, respectively. Thus, the unit costs $avg\_d(2) = 5.4$, $avg\_d(3) = 9$, $avg\_d(4) = 13.5$, and pred(2)=1, pred(3)=pred(4)=2. Next, we scan arc (1,4) and calculate $d(4) = 8y$, $x_4 = y$, and $avg\_d(4) = 8$. Since 8 is smaller than 13.5, we update node 4 by $d(4) = 8y$, $avg\_d(4) = 8$ and pred(4)=1. Now, node 4 becomes permanently labeled.

Repeating same steps for permanently labeled nodes, we can obtain $d(7) = 15y$, $x_7 = y$, $avg\_d(7) = 15$, and trace the predecessors $7 \rightarrow 4 \rightarrow 1$ for the path (1,4), (4,7). Details of this example is shown in Figure 3.14.

3.3.5 Summary

Section 3.3 describes our one-to-one UMDCP$_2$ problem and gives its mathematical formulation that focuses on a main path with minimum total unit cost to a destination node. We use CPLEX to calculate the optimal solution of one-to-one UMDCP$_2$, and develop a modified Dijkstra's algorithm to solve it.

3.4 One-to-one UMDCP$_3$

Our third uncapacitated minimum distribution cost problem sends flows from an S-node to some T-nodes considering the flow value for each arc and the distribution ratios of D-nodes. Unlike UMDCP$_2$, this problem does not allow flow to be shipped to any dummy termination node connecting from an O-node. That is, all the flows must be sent to the requested T-nodes or some other T-nodes. Figure 3.15 is an example of UMDCP$_3$ and its solution.

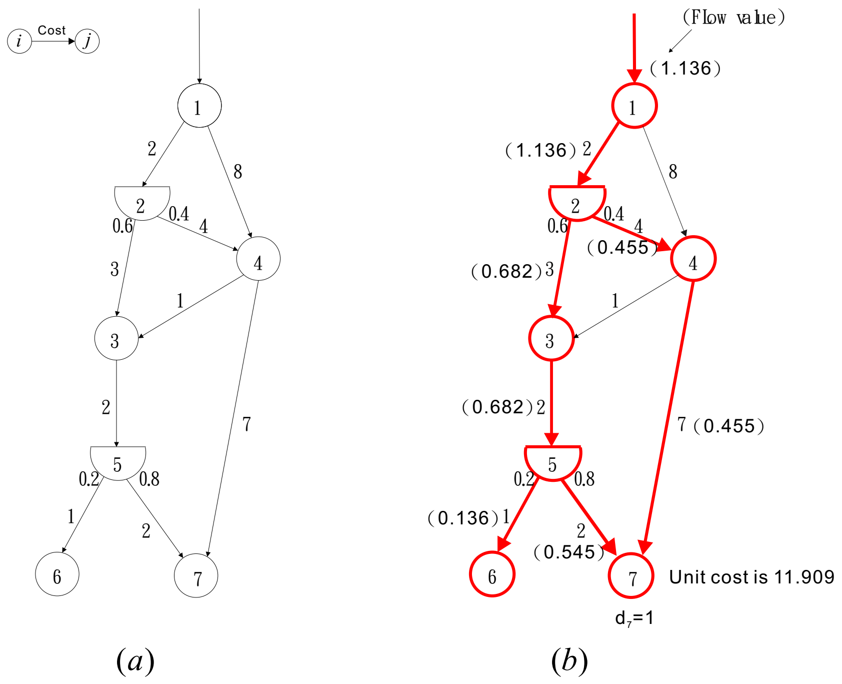

Figure 3.15: An UMDCP$_3$ problem

3.4.1 Formulation and Solution Method for a One-to-one UMDCP$_3$

Using the notations in Section 2.1, we can formulate a one-to-one UMDCP$_3$ as follows:

$$\begin{array}{ll} \min & \sum_{(i,j)\in A} c_{ij}x_{ij} \\ \textit{subject to} & \sum_{k\in L(i)} x_{ik} = \sum_{j\in E(i)} x_{ji} \quad \forall i \in N_O \cup N_D \\ & \sum_{(j,i)\in E(i)} x_{ji} \geq d_i \quad \forall i \in N_T \\ & x_{ik} = r_{ik}x_{ji} \quad \forall j \in E(i) \;\; \forall k \in L(i) \;\; \forall i \in N_D \end{array} \tag{3.4}$$

The objective function finds the minimum total cost for all arcs. The first constraint is the flow balance constraint for all O-nodes and D-nodes. The second one shows that all T-nodes must satisfy their demand. The third constraint represents the flow values passing through a D-node should obey its distribution ratios.

Unfortunately, we could not give an efficient algorithm to solve this linear program. To help future researchers to develop combinatorial algorithms, we give an enumeration method that finds all the feasible solutions (solution space) of the problem, and then chooses a solution that has the best objective value.

The solution space for a network problem maybe large. To shorten the time in searching solution space for a better solution more efficiently, we conclude that if flow passes through an O-node, a better solution will choose only one outgoing arc from that O-node to distribute the flow, instead of distributing flows on several outgoing arcs. The proof is shown in Appendix A. Based on this observation, we give a better enumeration algorithm.

3.4.2 Algorithm for Solving a One-to-one UMDCP$_3$

We develop an enumeration algorithm to solve our UMDCP$_3$ problem. The algorithm finds all subgraphs that connect the S-node and T-nodes based on Depthfirst search algorithm (DFS). After such a subgraph is identified, it will calculate the supply y and total cost of sending flows on the subgraph. Finally, it chooses the subgraph with minimum total unit cost as the optimal solution.

Let S and $\bar{S}$ be the sets of nodes, $S \cup \bar{S} = N$; aS and $a\bar{S}$ be the sets of arcs, $aS \cup a\bar{S} = A$. PATH-DATA is a structure for each node i in N, which include S, $\bar{S}$, aS, $a\bar{S}$, *path*, $tflag$, $dflag$, *pre_dflag*, etc. x_i and x_{ij} are the flow of each node i and each arc (i, j), r_{ij} is the distribution ratio of each outgoing arc (i, j) if node i is a D-node, c_{ij} is the cost for each arc (i, j). The steps of UMDCP$_3$ algorithm is shown as following with the detailed procedures in Appendix D.

Algorithm for solving UMDCP$_3$;

- ***Initialize***: All nodes are in $\bar{S}$ and all arcs are in $a\bar{S}$. $dflag$ of each node is 0 and *path* is $\emptyset$.
- ***step1***: If the source node s is an O-node, create a new PATH-DATA structure, and do O-**DFS(**s**)**.
- ***step2***: If the source node s is a D-node, create a new PATH-DATA structure for each node i out of s. Add node s to S and delete it from $\bar{S}$, add arc (s, i) to aS and delete it from $a\bar{S}$, add arc (s, i) to *path*. Do O-**DFS(**i**)**.
- ***step3***: Delete some paths which can not send flow to the specific termination node (i.e., delete those paths which doesn't be marked).
- ***step4***: Go to **Flow()**.

O-**DFS(**i**)**;

- ***step1***: If node i is in $\bar{S}$, add node i to S and delete it from $\bar{S}$.

- ***step2***: If the number of $A(i)$ is 0 and node i is the specific termination node, set $tflag$ of this $path$ is 1.

- ***step3***: If node i is a D-node, do D-**DFS**(i).

- ***step4***: If node i is an O-node and node j is in $\bar{S}$ for each arc (i, j), copy the $path$ of node i, add arc (i, j) to aS and delete it from $a\bar{S}$, add arc (i, j) to the $path$. If node j is an O-node, do O-**DFS**(j). If node j is a D-node, do D-**DFS**(j).

- ***step5***: If node i is an O-node and node j is in S for each arc (i, j), copy the $path$ of node i, add arc (i, j) to aS and delete it from $a\bar{S}$, add arc (i, j) to the $path$.

D-**DFS**(i);

- ***step1***: If node i is in $\bar{S}$, add node i to S and delete it from $\bar{S}$, let $pre\_dflag = dflag$ and $dflag = i$ of node i. Add arc (i, j) to aS and delete it from $a\bar{S}$, add arc (i, j) to the $path$ for each outgoing arc (i, j) of node i.

- ***step2***: If node j is the first node in $L(i)$, do O-**DFS**(j).

- ***step3***: Do O-**DFS**(j) for each remainder node j out of node i with $dflag = i$.

- ***step4***: If $dflag = i$, set $dflag = pre\_dflag$

Flow();

- ***step1***: For each path p, let n_p be the amount of nodes in this path, in_num be the number of incoming arcs of each node. Set $total\_cost = 0$, in_num of node s is 0, $flow$ of node s is x_s.

- ***step2***: If the *in_num* of node i is 0 for each node i in this path, add x_i and all $x_i r_{ik}$ out of node i. The *in_num* of node k should be reduced by one. Add *total_cost* and $c_{ik}x_i x_{ik}$ as new *total_cost*. Reduce n_p by one.

- ***step3***: Repeat step2 until $n_p = 0$.

3.5 Correctness of our One-to-one UMDCP$_3$ Algorithm

We give a one-to-one UMDCP$_3$ Algorithm by finding all possible feasible solutions, calculating each solution and choosing the best one. Since the method finds out all feasible solutions using a modified DFS algorithm, we discuss all possible cases that one may encounter in the searching process.

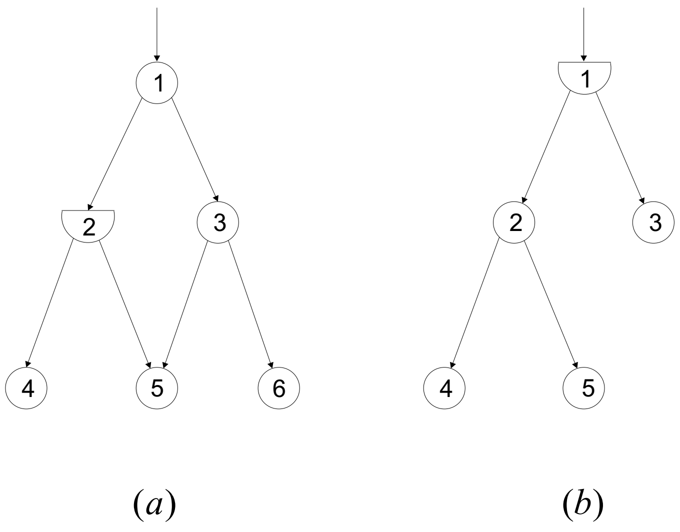

Figure 3.16: A general case of the modified DFS algorithm

Generally, for a network which contains O-nodes and D-nodes, if an O-node is selected, we do O-DFS algorithm, which is similar to the traditional DFS algorithm; otherwise, if a D-node is selected, then we do D-DFS algorithm, which considers all of the D-node's outgoing arcs and their following paths. Thus, all possible paths can be found. Figure 3.16(a) and (b) are both examples of a general case.

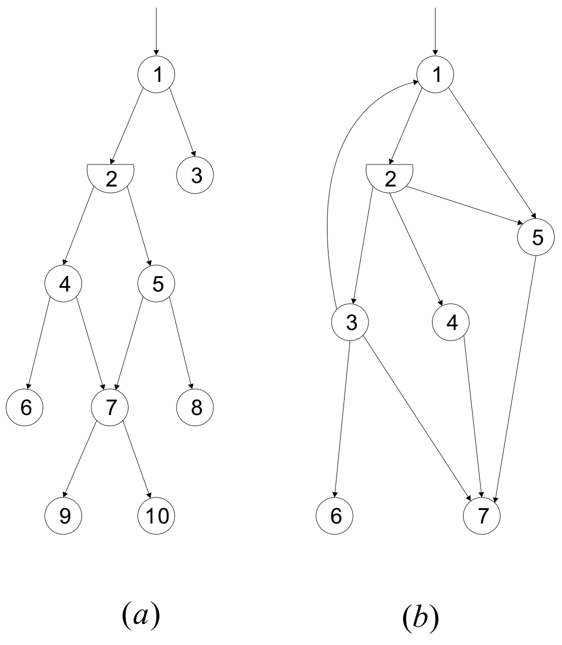

Figure 3.17: A gathering case of the modified DFS algorithm

Besides the general situation, there are two special cases we need to discuss. First, there may be more than one incoming arcs connecting to an O-node, i.e., there are two or more arcs with flows entering an O-node. When this happens, it must be resulted from some D-nodes upstream. That is, all outgoing arcs of a D-node are required to have flows to pass through, these flows may be gathered in some O-node later. Suppose an O-node is labeled from some D-node and later when this O-node is to be visited again, it must be from some D-node upstream. Since this O-node is searched previously, we do not need to do further search on this O-node. In Figure 3.17, (a) is a gathering case of the modified DFS algorithm. We want to send flow from node 1 to node 9. Suppose we search along the path (1,2), (2,4), (4,7), (7,9). Since node 2 is a D-node, we need to select arc (2,5). If we select arc (5,7), node 7 is visited again and flows are gathered in node 7. Thus, we stop and find a feasible subgraph with arcs (1,2), (2,4), (2,5), (4,7), (5,7), (7,9).

Second case is a "recycling" case. In particular, flows may go back to some visited nodes through other arcs. This situation would not appear in D-nodes, because each D-node has only one incoming arc. However, it may occur on O-nodes and then it is similar to a gathering case as firstly introduced above. One O-node may be visited several times, and we only need to search this O-node in the first time. Figure 3.17(b) shows the recycling case.

3.5.1 A One-to-one UMDCP$_3$ Example

Using the example illustrated in Figure 3.15(a), we show its formulation and the optimal solution by software. And, we use our algorithm to calculate the minimum total unit cost and find the shortest path.

In Figure 3.15(a), we want to find the shortest path from node 1 to node 7. Let y be the supply in node 1. The LP formulation of this example is illustrated as below:

$$\begin{aligned}
\min \quad & 2x_{12} + 8x_{14} + 3x_{23} + 4x_{24} + 2x_{35} + x_{43} + 7x_{47} + x_{56} + 2x_{57} \\
\textit{subject to} \quad & x_{12} + x_{14} = y \\
& x_{12} = x_{23} + x_{24} \\
& x_{23} + x_{43} = x_{35} \\
& x_{14} + x_{24} = x_{43} + x_{47} \\
& x_{35} = x_{56} + x_{57} \\
& x_{56} \geq 0 \\
& x_{57} + x_{47} \geq 1 \\
& x_{23} = 0.6x_{12} \\
& x_{56} = 0.2x_{35}
\end{aligned}$$

Then, we use CPLEX to solve this example. Using CPLEX, we obtain the optimal solution: $y^* = 1.136$, $x^*_{12} = 1.136$, $x^*_{23} = 0.682$, $x^*_{24} = 0.455$, $x^*_{35} = 0.682$, $x^*_{56} = 0.136$, $x^*_{57} = 0.545$, $x^*_{47} = 0.455$, and other variables are 0. Hence, we have the shortest path: (1,2), (2,3), (2,4), (3,5), (5,6), (5,7), (4,7) with the supply y equal to 1.136, and the minimum total unit cost from node 1 to node 7 equal to 11.909.

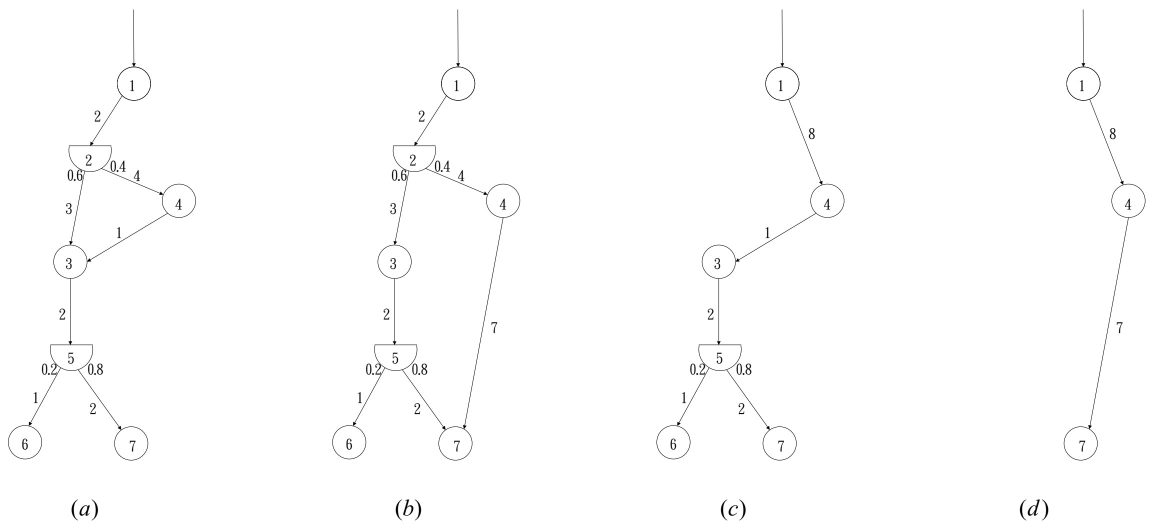

Figure 3.18: The feasible solution space using modified DFS algorithm

Then, we solve the same example by our UMDCP$_3$ algorithm based on our enumeration method. Using the modified DFS algorithm, the paths connects node 1 and node 7 are shown in Figure 3.18. Calculating the total cost, the flow value entering node 7, and total unit cost for each path in Figure 3.18, we have the shortest path in Figure 3.18(b): (1,2), (2,3), (2,4), (3,5), (5,6), (5,7), (4,7) with the optimal objective value equal to 11.909, the supply y equal to 1.136, and $x^*_{12} = 1.136$, $x^*_{23} = 0.682$, $x^*_{24} = 0.455$, $x^*_{35} = 0.682$, $x^*_{56} = 0.136$, $x^*_{57} = 0.545$, $x^*_{47} = 0.455$.

3.5.2 Summary

Section 3.4 describes our UMDCP$_3$ problem and its mathematical formulation that focuses on a main path with minimum total unit cost to a termination node where all the flows have to be shipped to T-nodes. We use CPLEX to calculate the optimal solution for one-to-one UMDCP$_3$. Then, we modify the DFS algorithm based on enumeration to develop an algorithm for solving our one-to-one UMDCP$_3$.

CHAPTER IV

UNCAPACITATED MINIMUM DISTRIBUTION COST PROBLEM (ONE-TO-SOME CASES)

We have introduced the one-to-one cases for our three uncapacitated minimum distribution cost problems in Chapter 3. In this Chapter, we will extend the models to be one-to-some cases and give their formulations.

4.1 One-to-some UMDCP$_1$

The one-to-some UMDCP$_1$ differs from its one-to-one case introduced in Section 3.2 only in the number of requested destinations. In particular, the one-to-one case seeks the minimum total cost to connect a source node to only one requested destination node, while the one-to-some case seeks the minimum total cost to connect a source not to several requested destination nodes.

In UMDCP$_1$, we just only consider the dependency relationship between arcs. The one-to-some case of UMDCP$_1$ and its solution are shown in Figure 4.1.

4.1.1 Formulation for a One-to-some UMDCP$_1$

The notations and settings for the formulation of the one-to-some UMDCP$_1$ is very similar to its one-to-one case as described in Section 3.2.1. Instead of setting x_{ij} to be a binary variable for each arc (i, j), here we make x_{ij} to represent a nonnegative

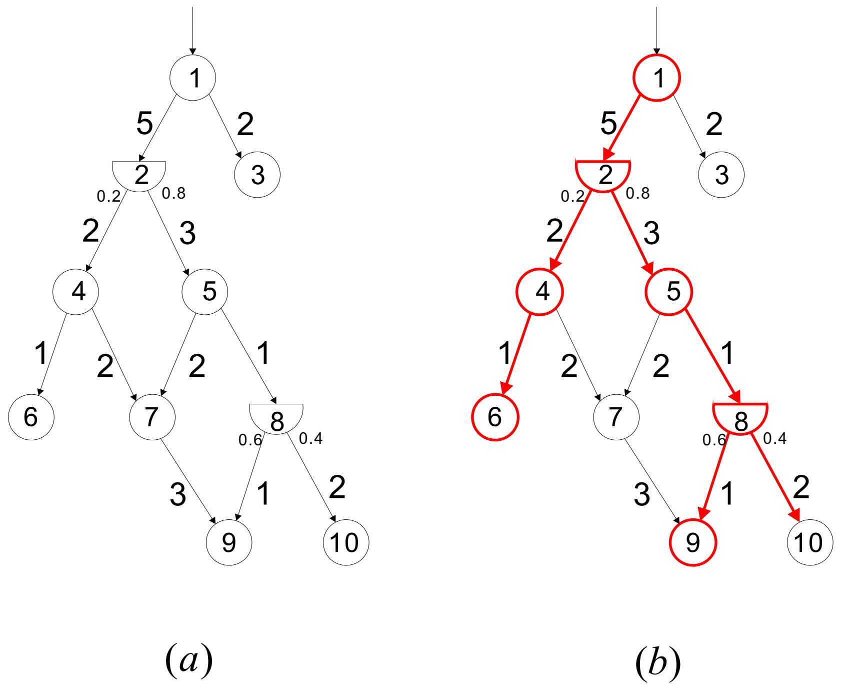

Figure 4.1: A one-to-some case of an UMDCP$_1$ problem

flow value and use a binary variable y_{ij} to represent whether arc (i, j) has positive flow or not. In particular, we set $y_{ij} = 1$ if $x_{ij} > 0$, and $y_{ij} = 0$ if $x_{ij} = 0$. Since now we have more than one destination to receive flow, it is possible that more original outgoing arcs from a D-node will appear in an optimal solution, unlike the one-to-one case where only one of these original outgoing arcs would appear in an optimal solution. For each outgoing arc of a D-node i, we also associate it with a parallel arc $(i, j)'$ with the same cost of the original arc (i, j). The purpose of arc $(i, j)'$ is already described in Section 3.2.1. Let B be the set of all arcs $(i, j)'$ for $i \in N_D$ and $j \in L(i)$. Suppose M is a very large number, we can formulate the one-to-some UMDCP$_1$ problem as follows:

$$\begin{array}{ll}
\min & \sum_{(i,j)\in A} c_{ij}y_{ij} + \sum_{(i,j)\in B} c_{ij}y'_{ij} \\
\textit{subject to} & \sum_{k\in L(i)} x_{ik} = \sum_{j\in E(i)} x_{ji} \quad \forall i \in N_O \\
& \sum_{j\in E(i)} x_{ji} \geq d(i) \quad \forall i \in N_T \\
& r_{ik}x_{ji} = x_{ik} + x'_{ik} \quad \forall j \in E(i) \;\; \forall k \in L(i) \;\; \forall i \in N_D \\
& x_{ij} - My_{ij} \leq 0 \quad \forall j \in A(i) \quad \forall i \in N \\
& x'_{ik} - My'_{ik} \leq 0 \quad \forall k \in L(i) \;\; \forall i \in N_D \\
& y_{ji} = y_{ik} + y'_{ik} \quad \forall j \in E(i), \;\; \forall k \in L(i), \;\; \forall i \in N_D \\
& \text{all } y_{ij} \text{ and } y'_{ij} \text{ are binary}
\end{array} \tag{4.1}$$

The objective function minimizes the total cost of all the selected arcs, including those parallel arcs. The first constraint is the flow balance constraint for all O-nodes. The second one shows that all T-nodes need to satisfy their demands. The third constraint represents flow distillation constraint. The forth and fifth constraints stand for the relationships between the flow values and whether an arc contains flow 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 $x_{ij} = 0$. The sixth one is relative to D-nodes: if the incoming arc of a D-node is selected, it is possible that only one or more than two original outgoing arcs can be selected.

4.1.2 A One-to-some UMDCP$_1$ Example

Taking the distribution network in Figure 4.1(a) for example, we want to send a unit of flow from the source node 1 to the specific termination nodes 6 and 9. For each D-node we associate parallel outgoing arcs. That is, arc $(2, 4)'$, $(2, 5)'$ for node 2, and arc $(8, 9)'$, $(8, 10)'$ for node 8. And, every arc (i, j) has a binary value y_{ij}.

The formulation for this example is shown as below:

$$\begin{aligned}
\min \quad & 5y_{12} + 2y_{13} + 2y_{24} + 2y'_{24} + 3y_{25} + 3y'_{25} + y_{46} \\
& + y_{47} + y_{57} + y_{58} + 3y_{79} + y_{89} + y'_{89} + 2y_{810} + 2y'_{810} \\
\textit{subject to} \quad & x_{12} + x_{13} = y \ , \quad x_{24} = x_{46} + x_{47} \\
& x_{25} = x_{57} + x_{58} \ , \quad x_{47} + x_{57} = x_{79} \\
& x_{46} \geq 1 \ , \quad x_{79} + x_{89} \geq 1 \ , \quad x_{810} \geq 0 \\
& 0.2x_{12} = x_{24} + x'_{24} \ , \quad 0.8x_{12} = x_{25} + x'_{25} \\
& 0.6x_{58} = x_{89} + x'_{89} \ , \quad 0.4x_{58} = x_{810} + x'_{810} \\
& x_{12} - My_{12} \leq 0 \ , \quad x_{13} - My_{13} \leq 0 \ , \quad x_{24} - My_{24} \leq 0 \\
& x_{25} - My_{25} \leq 0 \ , \quad x_{46} - My_{46} \leq 0 \ , \quad x_{47} - My_{47} \leq 0 \\
& x_{57} - My_{57} \leq 0 \ , \quad x_{58} - My_{58} \leq 0 \ , \quad x_{79} - My_{79} \leq 0 \\
& x_{89} - My_{89} \leq 0 \ , \quad x_{810} - My_{810} \leq 0 \\
& x'_{24} - My'_{24} \leq 0 \ , \quad x'_{25} - My'_{25} \leq 0 \\
& x'_{89} - My'_{89} \leq 0 \ , \quad x'_{810} - My'_{810} \leq 0 \\
& y_{12} = y_{24} + y'_{24} \ , \quad y_{12} = y_{25} + y'_{25} \\
& y_{58} = y_{89} + y'_{89} \ , \quad y_{58} = y_{810} + y'_{810} \\
& \text{all } y_{ij} \text{ and } y^{'}_{ij} \text{ are binary}
\end{aligned}$$

Using CPLEX, we obtain the selected arcs by $y^*_{12} = 1$, $y^*_{24} = 1$, $y^*_{25} = 1$, $y^*_{46} = 1$, $y^*_{58} = 1$, $y^*_{89} = 1$, $y^{'*}_{810} = 1$. Thus, the shortest path from node 1 to node 6 and 9 contains (1,2), (2,4), (4,6), (2,5), (5,8), (8,9), (8,10) with the total minimum cost is 15.

4.1.3 Comparison on One-to-one and One-to-some UMDCP$_1$ Solutions

We observe that a one-to-some UMDCP$_1$ optimal solution can not be composed of several one-to-one UMDCP$_1$ optimal solution.

Taking Figure 4.2(a) for example. Suppose we want to find the shortest path from node 1 to node 5 and 7. If we regard it as two one-to-one cases, i.e., the problems

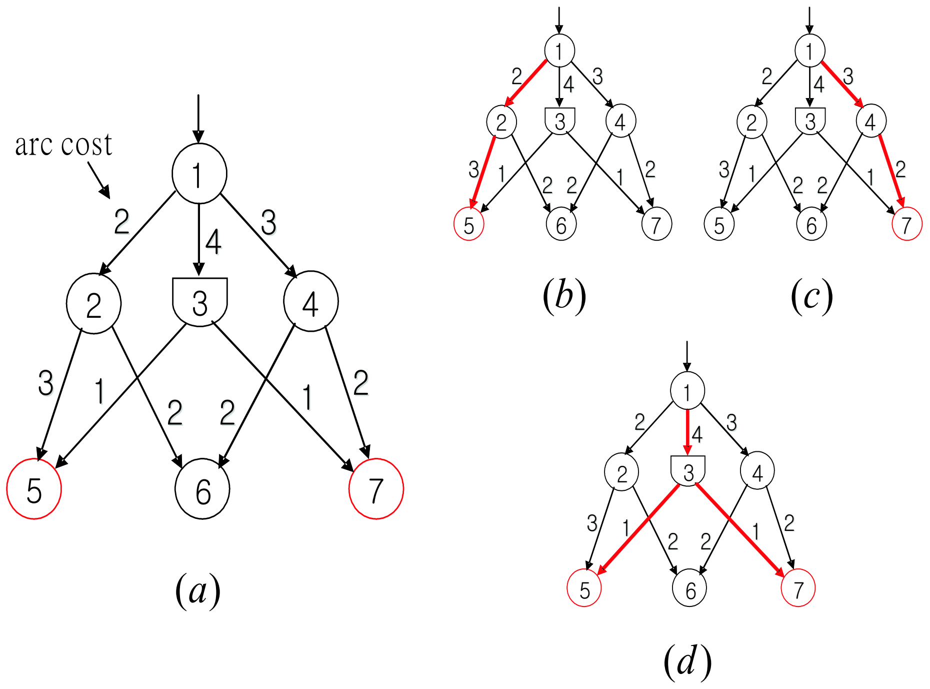

Figure 4.2: The one-to-some case of UMDCP$_1$

of sending flows from node 1 to node 5 and sending flows from node 1 to node 7 at the same time. Then, using our UMDCP$_1$ algorithm to solve these two one-to-one cases, we have the solutions as shown in Figure 4.2(b) and (c), respectively. However, this composed solution is different from the optimal one-to-some UMDCP$_1$ solution as shown in Figure 4.2(d). Hence, we know that the algorithm used to solve the one-to-one case of UMDCP$_1$ can not be directly used to solve the one-to-some case of UMDCP$_1$.

4.1.4 A Reduced Method for a One-to-some UMDCP$_1$

The one-to-some UMDCP$_1$ can also be reduced to an equivalent problem with simpler structure which only contains O-nodes as proposed in Section 3.2.5. The difference is that we use a binary variable y_{ij} to represent whether an arc (i, j) contains flow x_{ij} or not. After the transformation, we still can not solve the one-to-some UMDCP$_1$ by Dijkstra's algorithm as explained in previous Section. However, with a simpler problem structure, the transformed one-to-some UMDCP$_1$ can be solved by CPLEX in shorter computational time.

The simplified formulation is as following:

$$
\begin{array}{ll}
\min & \sum_{(i,j)\in A} c_{ij}y_{ij} \\
subject\ \ to & \sum_{k\in L(i)} x_{ik} = \sum_{j\in E(i)} x_{ji} \quad \forall i \in N_O \\
 & \sum_{j\in E(i)} x_{ji} = d(i) \quad \forall i \in N_T \qquad\qquad (4.2) \\
 & x_{ij} - My_{ij} \leq 0 \quad \forall j \in A(i) \quad \forall i \in N \\
 & \text{all } y_{ij} \text{ are binary}
\end{array}
$$

We also take Figure 4.1(a) for example, its original and reduced distribution networks are shown in Figure 4.3.

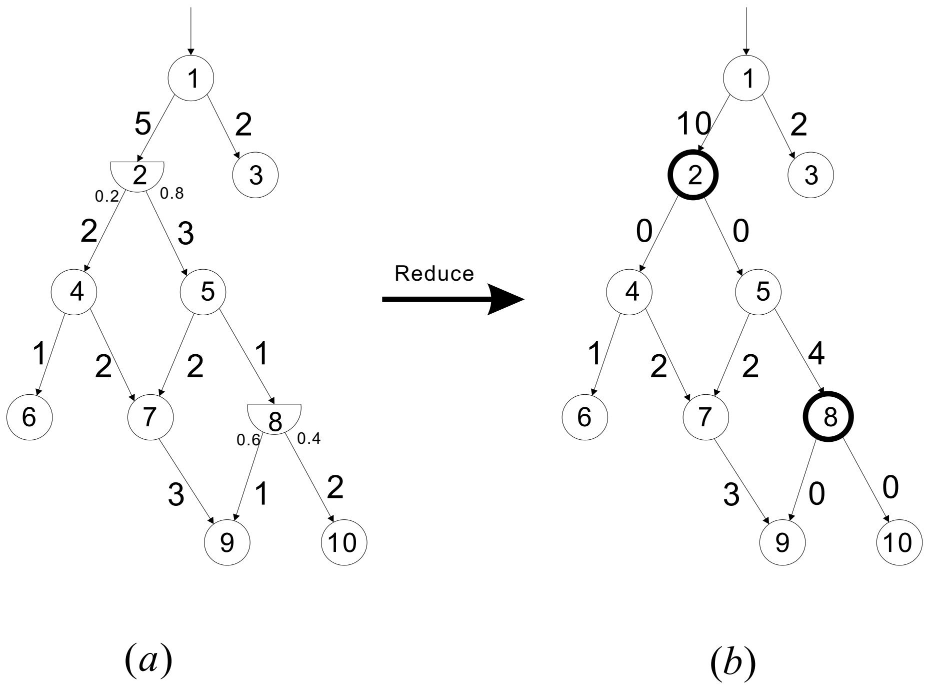

Figure 4.3: The original and reduced UMDCP$_1$ problems

We can formulate this example as following:

$$\begin{aligned}
\min \quad & 10y12 + 2y_{13} + y_{46} + 2y_{47} + 2y_{57} + 4y_{58} + 3y_{79} \\
\textit{subject to} \quad & x_{12} + x_{13} = 2 \\
& x_{12} = x_{24} + x_{25} \\
& x_{24} = x_{46} + x_{47} \\
& x_{25} = x_{57} + x_{58} \\
& x_{47} + x_{57} = x_{79} \\
& x_{58} = x_{89} + x_{810} \\
& x_{13} = 0 \\
& x_{46} = 1 \\
& x_{79} + x_{89} = 1 \\
& x_{810} = 0 \\
& \text{all } y_{ij} \text{ are binary}
\end{aligned}$$

Using CPLEX, we have the optimal solution: $y^*_{12} = 1, y^*_{24} = 1, y^*_{25} = 1, y^*_{46} = 1, y^*_{58} = 1, y^*_{89} = 1$. Because node 8 is a D-node, arc $(8, 10)$ must be selected. Hence, the shortest path from node 1 to node 6 and 9 contains arcs (1,2), (2,4), (4,6), (2,5), (5,8), (8,9), (8,10), and the objective value is 15. This result is the same as the example illustrated in Section 4.1.2.

4.2 One-to-some UMDCP$_2$

The one-to-some UMDCP$_2$ differs from its one-to-one case introduced in Section 3.3 only in the demands of the requested destinations. In particular, the one-to-one case seeks the minimum total cost to satisfy a unit demand for only one requested destination node, while the one-to-some case seeks the minimum total cost to satisfy a unit demand for each requested destination node. A one-to-some UMDCP$_2$ example and its solution are shown in Figure 4.4.

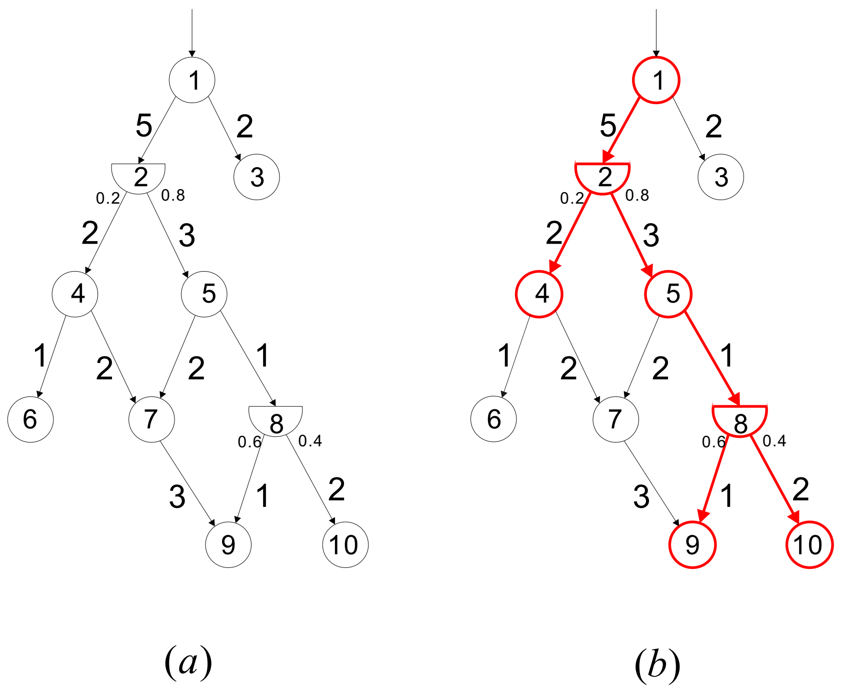

Figure 4.4: An one-to-some case of an UMDCP$_2$ problem

4.2.1 Formulation for a One-to-some UMDCP$_2$

The one-to-some UMDCP$_2$ formulation is almost the same as its one-to-one case except here we do not restrict only one original arc (i, j) out of a D-node i to appear in an optimal solution as the one-to-one case does. The one-to-some UMDCP$_2$ can be formulated as follows:

$$\begin{array}{lll}
\min & \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{(i,j)\in B} c_{ij}x'_{ij} & \\
\textit{subject to} & \sum_{k\in L(i)} x_{ik} = \sum_{j\in E(i)} x_{ji} \quad \forall i \in N_O & \\
& \sum_{(j,i)\in E(i)} x_{ji} \geq d_i \quad \forall i \in N_T & \\
& r_{ik}x_{ji} = x_{ik} + x'_{ik} \quad \forall j \in E(i) \;\; \forall k \in L(i) \;\; \forall i \in N_D & \\
& x_{ji} - My_{ji} \leq 0 \quad \forall j \in E(i) \;\; \forall i \in N_D & (4.3) \\
& x_{ik} - My_{ik} \leq 0 \quad \forall k \in L(i) \;\; \forall i \in N_D & \\
& x'_{ik} - My'_{ik} \leq 0 \quad \forall k \in L(i) \;\; \forall i \in N_D & \\
& y_{ji} = y_{ik} + y'_{ik} \quad \forall j \in E(i), \;\; \forall k \in L(i) \;\; \forall i \in N_D & \\
& \text{all } y_{ij} \text{ and } y'_{ij} \text{ are binary} &
\end{array}$$

The objective function finds the minimum total cost for all arcs in G', including those parallel arcs in the network. The first constraint is the flow balance constraint for all O-nodes. The second one shows that all T-nodes must satisfy their demand at least. The third constraint represents the flows on D-nodes' outgoing arcs should obey the distribution ratios. The forth, fifth, and sixth constraints 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 $x_{ij} = 0$. The seventh one is relative to D-nodes, we don't have the constraint $\sum_{(i,k)\in L(i)} y_{ik} \leq 1 \quad \forall i \in N_D$ because some original arcs may be chosen not just only one. And if some original arcs are selected, then all other parallel arcs from the same D-node will also be selected. $y_{ij} = 1$ or $y'_{ij} = 1$ means that arc (i, j) is selected, otherwise $y_{ij} = y'_{ij} = 0$ means arc (i, j) is not selected.

4.2.2 A One-to-some UMDCP$_2$ Example

We take Figure 4.4(a) for example, give its formulation and the optimal solution by using software.

In Figure 4.4(a), we want to find the shortest path from node 1 to node 9 and 10. Let y be the initial supply in node 1 and M is a very big number. The formulation can be formed as follow:

$$
\begin{aligned}
\min \quad & 5x_{12} + 2x_{13} + 2x_{24} + 2x'_{24} + 3x_{25} + 3x'_{25} + x_{46} \\
& +2x_{47} + 2x_{57} + x_{58} + 3x_{79} + x_{89} + x'_{89} + 2x_{810} + 2x'_{810} \\
\textit{subject to} \quad & x_{12} + x_{13} = y \\
& x_{24} = x_{46} + x_{47} \\
& x_{25} = x_{57} + x_{58} \\
& x_{47} + x_{57} = x_{79} \\
& x_{13} \geq 0 \ , \quad x_{46} \geq 0 \\
& x_{79} + x_{89} \geq 1 \ , \quad x_{810} \geq 1 \\
& 0.2x_{12} = x_{24} + x'_{24} \ , \quad 0.8x_{12} = x_{25} + x'_{25} \\
& 0.6x_{58} = x_{89} + x'_{89} \ , \quad 0.4x_{58} = x_{810} + x'_{810} \\
& x_{12} - My_{12} \leq 0 \ , \quad x_{58} - My_{58} \leq 0 \\
& x_{24} - My_{24} \leq 0 \ , \quad x'_{24} - My'_{24} \leq 0 \\
& x_{25} - My_{25} \leq 0 \ , \quad x'_{25} - My'_{25} \leq 0 \\
& x_{89} - My_{89} \leq 0 \ , \quad x'_{89} - My'_{89} \leq 0 \\
& x_{810} - My_{810} \leq 0 \ , \quad x'_{810} - My'_{810} \leq 0 \\
& y_{12} = y_{24} + y'_{24} \ , \quad y_{12} = y_{25} + y'_{25} \\
& y_{58} = y_{89} + y'_{89} \ , \quad y_{58} = y_{810} + y'_{810} \\
& \text{all } y_{ij} \text{ and } y'_{ij} \text{ are binary}
\end{aligned}
$$

Feeding the formulation into CPLEX, we calculate the optimal solution: $y^* = 3.125$, $x^*_{12} = 3.125$, $x^*_{24} = 0.625$, $x^*_{25} = 2.5$, $x^*_{58} = 2.5$, $x^*_{89} = 1.5$, $x^*_{810} = 1$, and other variables are 0. Hence, we have the shortest path: (1,2), (2,4), (2,5), (5,8), (8,9), (8,10), and the minimum total unit cost from node 1 to node 9 and 10 is 30.375.

4.2.3 Comparison on One-to-one and One-to-some UMDCP$_2$ Solutions

Like the discussion in Section 4.1.3, we can not treat a one-to-some UMDCP$_2$ as a composition of several one-to-one UMDCP$_2$.

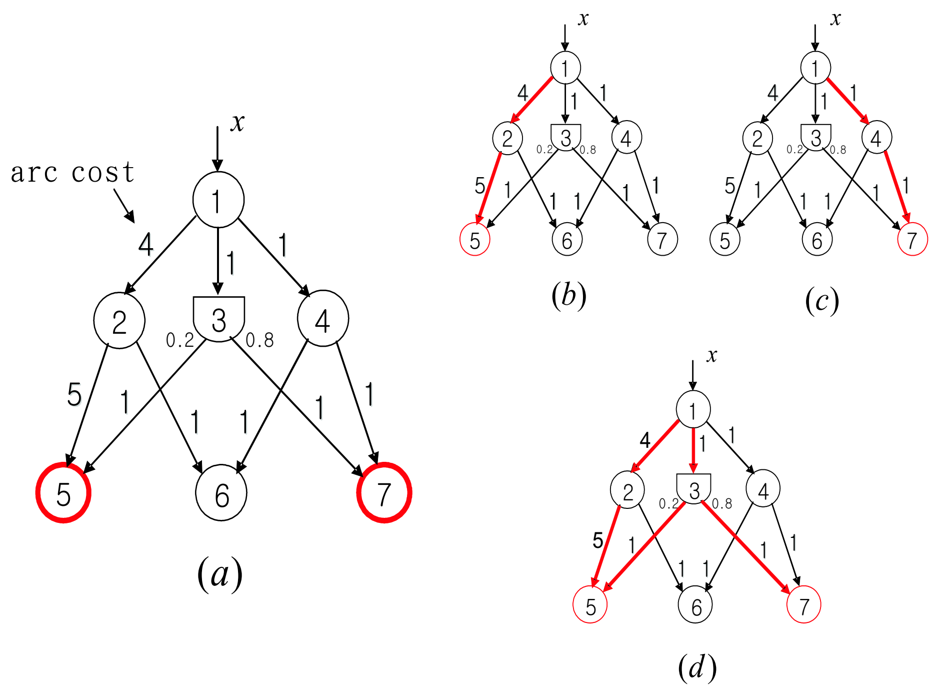

Figure 4.5: The one-to-some case of UMDCP$_2$

See Figure 4.5(a) for example. Suppose we want to find the shortest path from node 1 to node 5 and 7. If we solve the one-to-one UMDCP$_2$ from node 1 to node 5, the optimal solution is shown in Figure 4.5(b) and the optimal path for the one-to-one UMDCP$_2$ from node 1 to node 7 is shown in Figure 4.5(c),respectively. Thus if we combine these two one-to-one UMDCP$_2$ optimal solutions, we will get an optimal objective value to be $9 + 2 = 11$.

However, if we solve this one-to-some UMDCP$_2$ from node 1 to node 5 and node 7 at the same time, the optimal solution is shown in Figure 4.5(d) with optimal objective value equal to $4 \cdot 0.75 + 5 \cdot 0.75 + 1 \cdot 1.25 + 1 \cdot (0.2 \cdot 0.25) + 1 \cdot (0.8 \cdot 1) = 9.25 \neq 11$. Therefore, we can not use the one-to-one UMDCP$_2$ algorithm to solve the one-to-some UMDCP$_2$.

4.3 One-to-some UMDCP$_3$

The one-to-some UMDCP$_3$ differs from its one-to-one case introduced in Section 3.4 only in the demands of the requested destination nodes. In particular, the one-to-one case seeks the minimum total cost to satisfy a unit demand for only one requested destination node, while the one-to-some case seeks the minimum total cost to satisfy a unit demand for each requested destination node. A one-to-some UMDCP$_3$ example is shown in Figure 4.6.

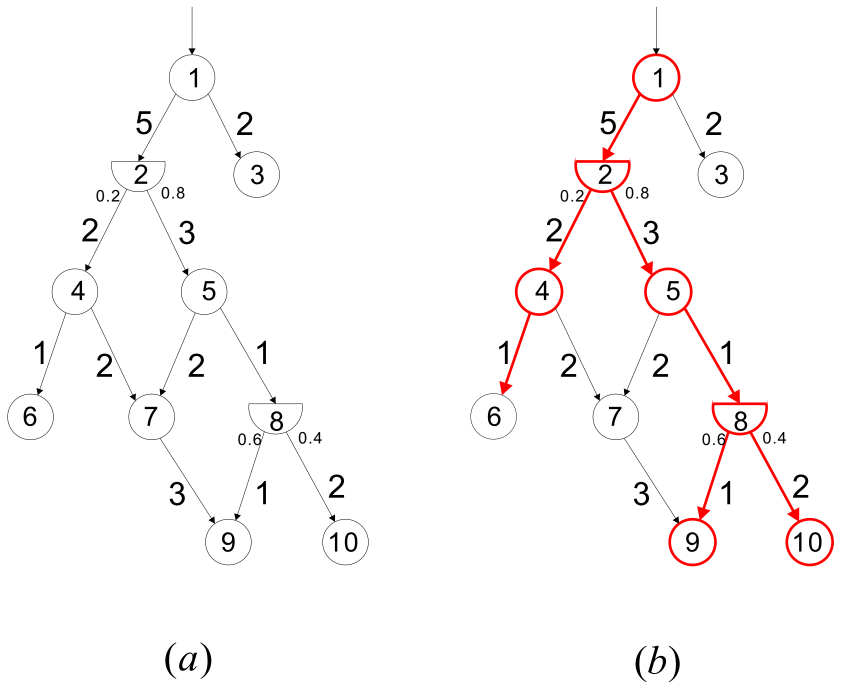

Figure 4.6: An one-to-some case of an UMDCP$_3$ problem

4.3.1 Formulation for a One-to-some UMDCP$_3$

The one-to-some UMDCP$_3$ has the same formulation as its one-to-one case as follows:

$$
\begin{array}{ll}
\min & \sum_{(i,j)\in A} c_{ij}x_{ij} \\
subject \;\; to & \sum_{k\in L(i)} x_{ik} = \sum_{j\in E(i)} x_{ji} \quad \forall i \in N_O \cup N_D \\
& \sum_{(j,i)\in E(i)} x_{ji} \geq d_i \quad \forall i \in N_T \\
& x_{ik} = r_{ik}x_{ji} \quad \forall j \in E(i) \;\; \forall k \in L(i) \;\; \forall i \in N_D
\end{array}
\tag{4.4}
$$

4.3.2 A One-to-some UMDCP$_3$ Example

Taking Figure 4.6(a) for example, we can give its formulation and the optimal solution using CPLEX. Note that in Figure 4.6(a), we want to find the shortest paths from node 1 to node 9 and 10, and the flows have to be shipped to the T-nodes. Let y be the initial supply in node 1. We can formulate a one-to-some UMDCP$_3$ example as follows:

$$
\begin{array}{ll}
\min & 5x_{12} + 2x_{13} + 2x_{24} + 3x_{25} + x_{46} + 2x_{47} + 2x_{57} \\
 & +x_{58} + 3x_{79} + x_{89} + 2x_{810} \\
\textit{subject to} & x_{12} + x_{13} = y \\
 & x_{12} = x_{24} + x_{25} \\
 & x_{24} = x_{46} + x_{47} \\
 & x_{25} = x_{57} + x_{58} \\
 & x_{47} + x_{57} = x_{79} \\
 & x_{58} = x_{89} + x_{810} \\
 & x_{13} \geq 0 \ , \quad x_{46} \geq 0 \\
 & x_{79} + x_{89} \geq 1 \ , \quad x_{810} \geq 1 \\
 & 0.2x_{12} = x_{24} \ , \quad 0.6x_{58} = x_{89}
\end{array}
$$

Using CPLEX, we can calculate the optimal solution: $y^* = 3.125$, $x^*_{12} = 3.125$, $x^*_{24} = 0.625$, $x^*_{46} = 0.625$, $x^*_{25} = 2.5$, $x^*_{58} = 2.5$, $x^*_{89} = 1.5$, $x^*_{810} = 1$, and other variables to be 0. Hence, the shortest paths contains arcs (1,2), (2,4), (4,6), (2,5), (5,8), (8,9), (8,10), and the minimum total unit cost from node 1 to node 9 and 10 is 31.

4.3.3 Comparison on One-to-one and One-to-some UMDCP$_3$ Solutions

Again, like the discussion in Section 4.1.3, we can not treat a one-to-some UMDCP$_3$ as a composition of several one-to-one UMDCP$_3$.

For example, suppose we want to find the shortest path from node 1 to node 5 and 7 in Figure 4.5(a). If we solve two one-to-one UMDCP$_3$ problems from node 1 to node 5 and from node 1 to node 7, respectively, we will obtain the optimal solutions that use arcs (1,2), (2,5) for the first case and arcs (1,4), (4,7) for the second case. However, if we solve the one-to-some UMDCP$_3$ from node 1 to node 5 and node 7 at the same time, the optimal solution will use arcs (1,2), (2,5), (1,3), (3,5), (3,7) with

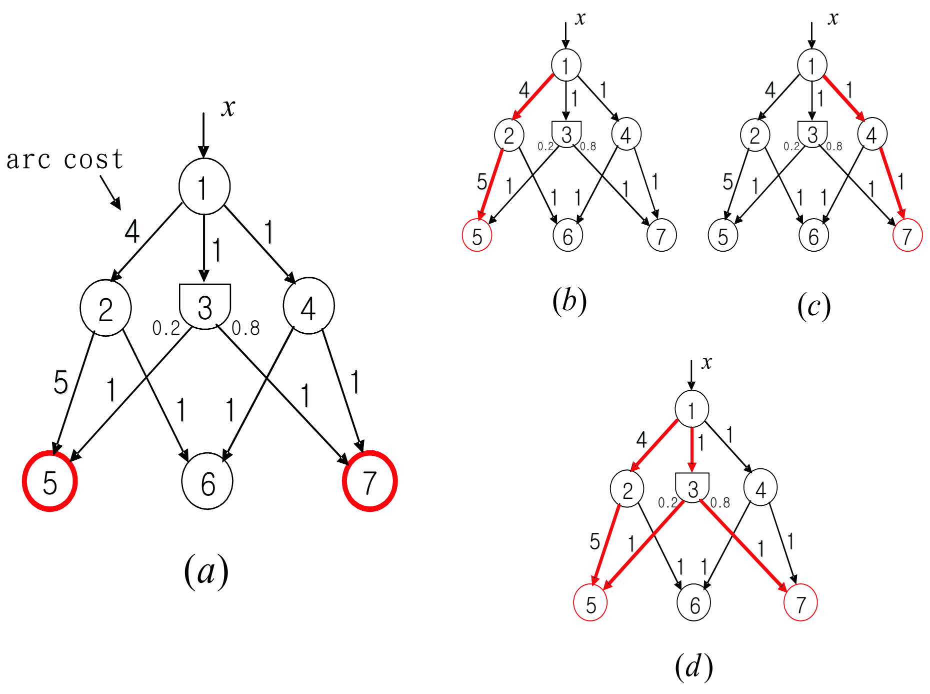

Figure 4.7: The one-to-some case of UMDCP$_3$

the total cost 9.25. Thus, we can not solve the one-to-some UMDCP$_3$ directly by using a one-to-one UMDCP$_3$ algorithm.

4.4 Summary

This Chapter describes our three one-to-some UMDCP problems and gives their mathematical formulations that focus on a main path with minimum total cost to connect the source node to several destination nodes, or to satisfy unit demands for several requested destination nodes. We give counter examples to illustrate that one-to-one solutions can not be used to solve the one-to-some problems. At this moment, there are still no efficient combinatorial algorithms to solve the one-to-some UMDCP problems. One still has to formulate the one-to-some UMDCPs and use the optimization software such as CPLEX to solve all the one-to-some UMDCPs.

CHAPTER V

CONCLUSIONS AND FUTURE DIRECTIONS

5.1 Summary and Contributions

Extending the MNF model presented by Fang and Qi (2003), this thesis introduces three uncapacitated minimum distribution cost problems (UMDCP), proposes their formulations and develops algorithms to solve them. Since all of our UMDCPs have no arc capacity constraints, we treat them as specialized shortest path problems with side constraints called flow distillation constraints. We summarize this thesis and mention our contributions as below:

1. The UMDCP$_1$ finds the minimum cost to connect an S-node to several T-nodes. 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; while for its one-to-some case, several outgoing arcs may further connect to other arcs when necessary. In UMDCP$_1$, the cost is related to whether an arc is in a solution or not, and we ignore the cost of flows. In this problem, the solution is a subgraph that contains paths connecting the S-node and all the requested T-nodes, together with some side branch arcs that are side-products of some D-nodes selected in the paths. We have proposed two mathematical formulations for

both the one-to-one and one-to-some UMDCP$_1$. For its one-to-one case, we can transform it to be a conventional shortest path problem and solve it by Dijkstra's algorithm.

2. The UMDCP$_2$ finds the minimum cost to receive a unit flow for each requested T-nodes from an 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 only allows only one of the arcs outgoing from a selected D-node to connect further to other arcs in its one-to-one case while it allows more such arcs when necessary in its one-to-some case. However, unlike UMDCP$_1$ that considers only the dependency relations between arcs, UMDCP$_2$ focuses on the total unit flow costs for all its requested destinations. We have proposed mathematical formulations for both the one-to-one and one-to-some UMDCP$_2$. For its one-to-one case, we give a modified Dijkstra's algorithm and show its correctness.

3. The UMDCP$_3$ is the most general and firstly presented in the paper by Fang and Qi (2003). Similar to UMDCP$_2$, UMDCP$_3$ also considers flow costs and focuses on minimizing the total unit costs for the requested destinations. However, UMDCP$_3$ forces all the outgoing arcs of a selected D-node to connect further to other arcs, unlike UMDCP$_2$ which only does it when necessary. That is, all the flows must be sent to the requested T-nodes or some other T-nodes. We have proposed a mathematical formulation for both the one-to-one and one-to-some UMDCP$_1$. Unfortunately we could not give efficient combinatorial algorithm for its one-to-one case. Instead, we give an enumeration method which enumerates all subgraphs connecting the requested source and destination nodes. Our enu-

meration method can be useful for future research since other related problems such as the max-flow or min-cost distribution network flow problems may use our enumeration method to test their solutions for correctness and efficiency.

4. In this thesis, 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 may offer better managerial insights.

5.2 Future Research Directions

We have encountered several difficult and challenging problems. We list several ones that we consider to be important for future researchers to investigate as follows:

1. Development of efficient combinatorial algorithm for solving the one-to-one UMDCP$_3$. Our enumeration method can be used for verification but is not efficient. We would be pleased to see someone can overcome the difficulty of counting the exponentially combinations of paths connecting D-nodes to the requested destinations.

2. Development of efficient combinatorial algorithm for solving the all one-to-some UMDCPs.

   So far we are only able to give their correct mathematical formulations. We give counter examples to show that the one-to-some cases can not be solved by iteratively application of the one-to-one UMDCP solution methods. This observations should discourage those who attempts to use conventional shortest path algorithms for solving the one-to-some cases.

3. In this thesis we only considered the appearance of D-nodes and O-nodes. It

would be interesting to include the C-nodes together with the I-nodes as introduced by Fang and Qi (2003). Such a problem should be more general and challenging and worth investigation.

REFERENCES

Ahuja, R. K., Magnanti, T. L. and Orlin, J. B. *Network flows: Theory, algorithms, and applications.* Prentice-Hall, New Jersey, 1993.

Ahuja, R. K., Mehlhorn, K., Orlin, J. B. and Tarjan, R. E. Faster algorithms for the shortest path problem. *Journal of ACM*, **37**, 213-223, 1990.

Bellman, R. E. On a routing problem. *Quarterly of Applied Mathematics*, **16**, 87-90, 1958.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. *Introduction to algorithms, second edition.* The MIT Press, 2001.

Dial, R. Algorithm 360: Shortest path forest with topological ordering. *Communications of the ACM*, **12**(11), 632-633, 1969.

Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**, 269-271, 1959.

Fang, S. C. and Qi, L. Manufacturing network flows: A general network flow model for manufacturing process modelling. *Optimization Methods and Software*, **18**(2), 143-165, 2003.

Floyd, R. W. Algorithm 97: Shortest path. *Communications of the ACM*, **5**(6), 345, 1962.

Ford, L. R. *Network flow theory.* Santa Monica, California: The RAND Corp., 1956.

Fredman, M. L. and Tarjan, R. E. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, **34**(3), 596-615, 1987.

Johnson, E. L. 1972. On shortest paths and sorting. In *Proceedings of the acm 25th annual conference* (p. 510-517).

APPENDICES

APPENDIX A

Reduce the Feasible Solution Space for solving a one-to-one UMDCP

Lemma 1. *When an optimal one-to-one UMDCP solution passes an O-node, it will choose only one of its outgoing arcs to ship flows, even if it has more than one outoing arcs.*

Proof. There are two cases we need to discuss: (1) a network which only contains O-nodes; (2) a network contains O-nodes and D-nodes.

In the first network, all nodes are O-nodes and we obtain that the optimal solution is a path from the source node to the termination node, obviously. Thus, it is clear to know that if the flow goes through one O-node, it will choose one outgoing arc of the O-node. We know that an O-node can represent a constraint. That is, in these basis graphs, an arc corresponds to a node. An example is illustrated in Figure A.1(a). Hence, we have two feasible basis solutions in Figure A.1(b) and (c).

Next, we discuss the second situation which a network contains O-nodes and D-nodes. When we add D-nodes into a traditional network, if a D-node has k outgoing arcs, it will have k constraints. Thus, we obtain an O-node to represent a constraint and a D-node to stand for k constraints.

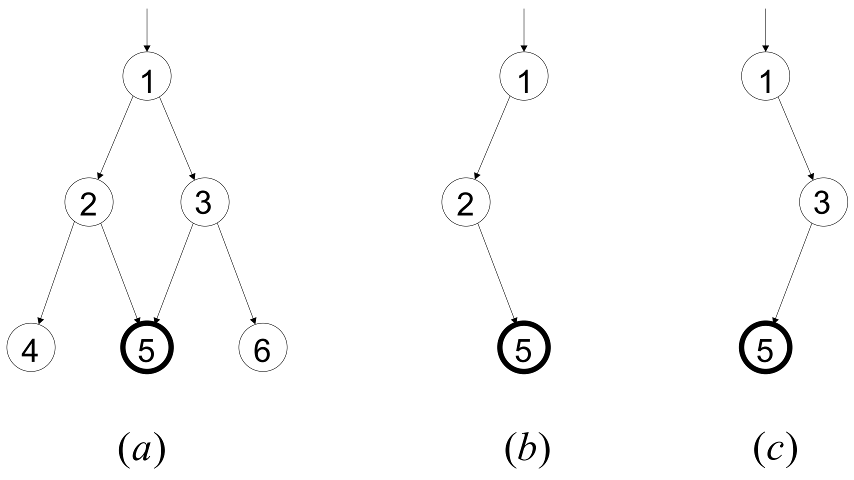

Figure A.1: A network contains O-nodes only

First, we consider a network with some O-nodes and one D-node as shown in Figure A.2(a). For these O-nodes, each of them represents one constraint and there is one arc corresponding to each O-node, respectively. In Figure A.2(b), it is the same as the feasible basic solution of a network with O-nodes only. In Figure A.2(c), the D-node represents two constraints, i.e., arc (1,3) and arc (3,6) correspond to D-node 3.

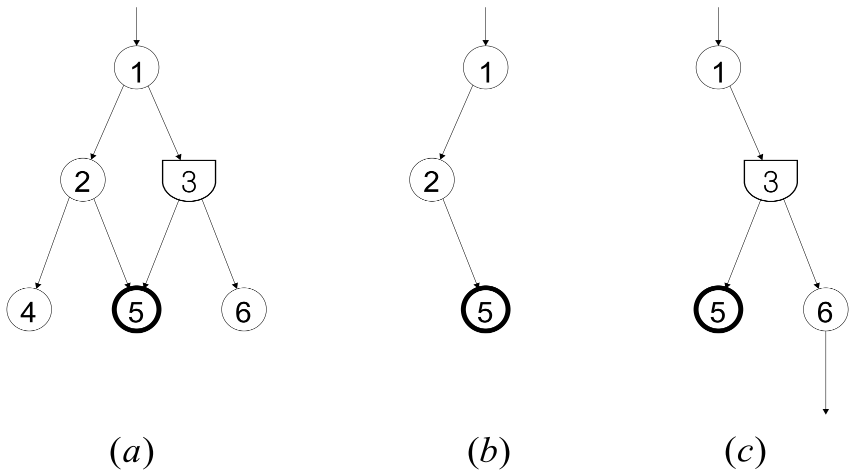

Figure A.2: A network contains O-nodes and D-nodes

When a network contains two D-nodes, the discussion is similar to a network with one D-node. Each O-node represents one constraint and has an arc corresponding to them. For each D-node, if a D-node has k outgoing arcs, there are k arcs corresponding to this D-node.

Therefore, we know that no matter a network contains no or more than one D-node, it can be discussed similarly. Furthermore, we observe that a feasible basic solution of a network has the following property: if the flow goes through one O-node, the flow will choose only one of its outgoing arcs to travel, even if there is more than one outgoing arcs. $\square$

APPENDIX B

Algorithm for solving UMDCP$_1$

```
Begin
    $S := \emptyset$; $\bar{S} := N$;
    $d(i) := \infty$ for each node $i \in N$;
    If $S \in N_O$ Then
        $d(s) := 0$ and pred$(s) := 0$;
    Else
        $d(s) := \sum_{(s,j) \in A(s)} c_{sj}$ and pred$(s) := 0$;
        $d(j) := d(s)$ and pred$(j) := s$ for each node $j \in L(s)$;
    While $|S| < n$ do
        let $i \in \bar{S}$ be a node for which $d(i) = \min\{\ d(j) : j \in \bar{S}\ \}$;
        $S := S \cup \{i\}$; $\bar{S} := \bar{S} - \{i\}$;
        If $i \in N_O$ Then
            For each arc $(i, j) \in A(i)$ do
                If $j \in N_D$ Then
                    $d(j) = d(i) + c_{ij} + \sum_{(j,k) \in A(j)} c_{jk}$ and pred$(j) := i$;
                    If $d(k) > d(j)$ Then
                        $d(k) := d(j)$ and pred$(k) := j$ for each node $k$;
                Else
                    If $d(j) > d(i) + c_{ij}$ Then
                        $d(j) := d(i) + c_{ij}$ and pred$(j) := i$;
    End While
End
```

APPENDIX C

Algorithm for solving UMDCP$_2$

```
Begin
    $S := \emptyset$; $\bar{S} := N$;
    $d(i) := \infty$ and $avg\_d(i) := \infty$ for each node $i \in N$;
    If $S \in N_O$ Then
        $d(s) := 0$, $avg\_d(s) := 0$, pred$(s) := 0$ and $x_s := x_s$;
        $x_{sj} := x_s$ for each arc $(s, j) \in A(s)$;
    Else
        $x_s := x_s$; $x_{sj} := r_{sj}x_s$ and $x_j := x_{sj}$ for each arc $(s, j) \in A(s)$;
        $d(s) := \sum_{(s,j)\in A(s)} c_{sj}x_{sj}$, $avg\_d(s) := \frac{d(s)}{x_s}$, pred$(s) := 0$;
        $d(j) := d(s)$, $avg\_d(j) := \frac{d(j)}{x_j}$, pred$(j) := s$ for each node $j \in L(s)$;
    While $|S| < n$ do
        let $i \in \bar{S}$ be a node for which $avg\_d(i) = \min\{\ avg\_d(j) : j \in \bar{S}\ \}$;
        $S := S \cup \{i\}$; $\bar{S} := \bar{S} - \{i\}$;
        If $i \in N_O$ Then
            For each arc $(i, j) \in A(i)$ do
                If $j \in N_D$ Then
                    $x_j := x_{ij}$;
                    $x_{jk} := r_{jk}x_{ij}$ and $x_k := x_{jk}$ for each arc $(j, k) \in A(j)$;
                    $d(j) = d(i) + c_{ij}x_{ij} + \sum_{(j,k)\in A(j)} c_{jk}c_{jk}$,
                    $avg\_d(j) := \frac{d(j)}{x_j}$, pred$(j) := i$;
                    If $avg\_d(k) > \frac{d(j)}{x_k}$ for each node $k \in L(j)$ Then
                        $d(k) := d(j)$, $avg\_d(k) := \frac{d(k)}{x_k}$,
                          and pred$(k) := j$ for each node $k \in L(j)$;
                Else
                    $x_j := x_{ij}$; $x_{jk} := x_{ij}$ and $x_k := x_{jk}$ for each arc $(j, k) \in A(j)$;
                    If $avg\_d(j) > \frac{d(i)+c_{ij}xij}{x_j}$ Then
                        $d(j) := d(i) + c_{ij}x_{ij}$, $avg\_d(j) := \frac{d(i)+c_{ij}x_{ij}}{x_j}$,
                          and pred$(j) := i$;
    End While
End
```

APPENDIX D

Algorithm for solving UMDCP$_3$

```
begin
    $S := \emptyset$, $\bar{S} := N$; $aS := \emptyset$, $a\bar{S} := N$; $path = \emptyset$ and $dflag = 0$;
    **If** $s \in N_O$ **Then do**
    **begin**
        Create a new PATH-DATA;
        $O$-DFS($s$);
    **end**
    **Else**
    **begin**
        **for** each node $i \in L(s)$ **do**
        **begin**
            Create a new PATH-DATA;
            $S := S \cup \{s\}$, $\bar{S} := \bar{S} - \{s\}$; $aS := aS \cup (s, i)$, $a\bar{S} := a\bar{S} - (s, i)$;
            $path := path \cup (s, i)$;
            **If** $i \in N_O$ **Then** $O$-DFS($i$);
            **Else** $D$-DFS($i$);
        **end**
    **end**
    Delete some paths $p$ which flow can't go to the specific termination node;
    Flow();
end
```

```
$O$-DFS($i$)
begin
    If $i \in \bar{S}$ Then do
    begin
        $S := S \cup \{i\}$; $\bar{S} := \bar{S} - \{i\}$;
        If $L(i) = \emptyset$ and $i \in N_T$ Then $tflag = 1$ and return;
        If $L(i) = \emptyset$ Then Return;
        If $i \in N_D$ Then $D$-DFS($i$);
        Else
        begin
            for each $j \in L(i)$ do
            begin
                If $j \in \bar{S}$ Then do
                begin
                    If $j \in L(i)$ is not last node Then copy current $path$ and push
$path$;
                    $aS := aS \cup (i, j)$; $a\bar{S} := a\bar{S} - (i, j)$;
                    $path := path \cup (i, j)$;
                    If $j \in N_O$ Then $O$-DFS($j$);
                    Else $D$-DFS($j$);
                end
                Else
                begin
                    If $j \in L(i)$ is not last node Then copy current $path$ and push
$path$;
                    $aS := aS \cup (i, j)$; $a\bar{S} := a\bar{S} - (i, j)$;
                    $path := path \cup (i, j)$;
                end
            If $j \in L(i)$ is not last node Then pop $path$;
            end
        end
    end
end
```

```
$D$-**DFS**($i$)
**begin**
    **If** $i \in \bar{S}$ **Then do**
    **begin**
        $S := S \cup \{i\}$; $\bar{S} := \bar{S} - \{i\}$;
        $pre\_dflag := dflag$;
        $dflag := i$;
        **for** each $j \in L(i)$ **do**
        **begin**
            $aS := aS \cup (i, j)$; $a\bar{S} := a\bar{S} - (i, j)$;
            $path := path \cup (i, j)$;
        **end**
        **If** $j \in L(i)$ is the first node **Then** $O$-DFS($j$);
        **for** each $j \in L(i)$ not be the first node **do**
        **begin**
            **for** each $path$ with $dflag = i$ **do**
                $O$-DFS($j$);
        **end**
        **for** each $path$ **do**
            **If** $dflag = i$ **Then** $dflag := pre\_dflag$;
    **end**
**end**
```

```
Flow()
begin
    for each path $p$ do
    begin
        let $n_p$ be the number of nodes on path $p$
        let $in\_num(i)$ be the number of incoming arcs of each node $i$ on path $p$;
        $total\_cost(p) := 0$; $in\_num(s) := 0$; $x_s := x_s$; $flow(p) := 0$;
        While $n_p > 0$ do
        begin
            for each node $i$ on path $p$ do
            begin
                If $in\_num(i) = 0$ Then do
                begin
                    for each node $k \in L(i)$ on path $p$ do
                    begin
                        $flow(p) := flow(p) + x_{ik}$;
                        $in\_num(k) := in\_num(k) - 1$;
                        $total\_cost(p) := total\_cost(p) + c_{ik}x_{ik}$;
                        $n_p = n_p - 1$;
                    end
                end
            end
        end
    end
    $unit\_cost(p) := \frac{total\_cost(p)}{flow(p)}$;
end
```