國　立　成　功　大　學

工業與資訊管理研究所

碩　士　論　文

以最小平方主對偶演算法求解網路問題

Solving the network flow problem by a least-squares primal-dual method

指導教授：王逸琳 博士

研究生：張正翰

中華民國九十五年八月

國立成功大學

碩士論文

Solving the network flow problem by a least-squares primal-dual method

研究生：張正翰

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

李宇欣 李宇欣

許瑞麟 許瑞麟

王逸琳 王逸琳

指導教授： 王逸琳

系(所)主管： 李賢得

中華民國 九十五 年 五 月 二十七 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(學院)工業與資訊管理系所

組 九十四 學年度第 二 學期取得 碩 士學位之論文。

論文名稱：Solving the network flow problem by a least-squares primal-dual method

☑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：張正華 學號：P36931027

（親筆正楷） （務必填寫）

日期:民國 95 年 8 月 24 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)

摘要

網路問題通常可被視為具有特殊限制式結構的線性規劃問題，因此一般用來求解線性規劃問題的方法，亦可被應用於求解網路問題。其中，「最小平方主對偶演算法」是最近被提出的一種線性規劃問題的有效解法，該演算法藉由多次地求解一最小非負平方的子問題來改善其對偶解，並可避免在求解過程中陷入退化解而停滯不前，因此可用比傳統的單體法 (simplex method) 更少的計算迴圈來求得最佳解。

本研究將根據「最小平方主對偶演算法」的概念，發展一套求解網路問題的圖形化演算法，並探討如何有效地取得初始對偶可行解，以及如何求解具有流量上限的最小流量成本問題。此外，我們亦將主問題與對偶問題的角色對調，提出一套「最小平方對偶主演算法」。並發現若以該演算法求解最大流量問題時，若將各弧上的單位流量成本視為弧上兩端點的電動勢，則原先所求解的非負最小平方子問題，可視為在一個內含二極體電路上求解其電流之問題。易言之，電路學中的柯西赫夫定律可被用來求解最小平方對偶主演算法中的非負最小平方和問題。因此，最大流量問題與最小流量成本問題皆可利用最小平方對偶主演算法，結合柯西赫夫定律，來避免退化過程並有效求解。

關鍵字：網路最佳化，最小成本流量問題，主對偶演算法，非負最小平方和，退化，最大流量問題

ABSTRACT

The network flow problem is a specialized Linear Programming problem (LP) due to its special constraint structure. An LP solution method may have a more efficient implementation when applied for solving the network flow problem. Recently, a new primal-dual algorithm called the least-squares primal-dual (LSPD) method has been proposed to solve LPs with good performance since it guarantees nondegenerate pivoting in each iteration. In each prmial-dual iteration, the LSPD algorithm solves a nonnegative least squares (NNLS) subproblem to obtain an improving direction for its dual variables.

In this thesis, we develop techniques related with the LSPD method for solving network flow problems. Issues regarding efficient ways to obtain an initial dual feasible solution and techniques to deal with capacitated network flow problems will be investigated. We also propose a new least-squares dual-primal (LSDP) algorithm which differs from the LSPD algorithm in exchanging the roles of the primal and dual formulations. When solving for the max-flow problem, the NNLS subproblem in our LSDP algorithm can be treated as an algorithm to calculate the current on an electrical network with diodes, where the unit-flow cost associated with an arc can be treated the electrical potential. Thus the Kirchhoff's law can be applied to solve the NNLS subproblem. Similar techniques can be applied to solve the MCF problem.

keywords: network optimization, minimum cost flow problem, primal-dual algorithm, nonnegative least-squares, degeneracy, maximum flow problem

ACKNOWLEDGEMENTS

首先感謝指導教授王逸琳老師在論文撰寫期間給予的細心指導，以及兩年來所給予的諸多良善的建言，培養我務實與追求完美的處事態度。口試期間感謝陳文智老師、李宇欣老師及許瑞麟老師對本論文提供寶貴的意見，使本論文得以更加完善。

感謝研究所的所有同學與我一同分享這段令人難忘的碩士生涯，因為有你們讓我這兩年除了研究以外還有更多美好的回憶。還有惠娥、修杰、橙坤、群達與姿君，研究室的夥伴們，因為有你們開朗的笑聲，讓我在研究所的日子更顯歡樂。慶幸有你們陪我度過這難忘的日子。

最後，要感謝我的父母與家人，沒有你們在家鄉的支持與鼓勵，就沒有今天的我。感謝你們這些年來對我的包容與關懷，僅將這份論文與碩士學位的榮耀獻給你們。

TABLE OF CONTENTS

LIST OF FIGURES

Figure

CHAPTER I

INTRODUCTION

1.1 Background and motivation

With the competitive global business, good supply chain management for a global corporation becomes the key factors for its success. Good supply chain management requires efficient commodity transportation planning throughout the logistics network flow system, which spurs more researchers in the field of network optimization to investigate more efficient solution methods.

Conventional network optimization problems consist of three major types, "*the shortest path problem*", "*the maximum flow problem*", and "*the minimum cost flow problem*". In particular, the shortest path problem finds a path with the shortest length from some specific source nodes to some specific terminal nodes where the length for a path is the sum of the arc lengths it passes. Depending on the number of source and terminal nodes, there are "one-to-one", "one-to-all", "all-to-one", "all-to-all", and "some-to-some" shortest path problems. The maximum flow problem focuses on how to transport the largest amount of flow through the network from source nodes to terminal nodes using capacitated arcs. While the shortest path problem considers only the arc length (i.e. unit flow cost passing through an arc) and the maximum flow problems considers only the arc capacities, the minimum cost flow (MCF) problem

considers both the arc length and capacity. In particular, the MCF problem studies on how to transport the required amount of flow through the network from source nodes to terminal nodes with minimum total cost, where each arc is associated with flow bound (upper and lower) constraints and a unit flow cost (arc length). In fact, both the shortest path problem and the maximum flow problem can be considered as specialized MCF problems and thus can be transformed into MCF problems. For example, a one-to-one shortest path problem can be viewed as an MCF problem with a unit flow for the requested origin-destination pair on a network with uncapacitated arcs. When we add an artificial arc with negative cost from the terminal node to the source node, we can also turn a maximum flow problem into an MCF problem. Figure 1.1 illustrates an MCF example where 3 units are to be shipped from node 1 to node 4 via 5 capacitated arcs in minimal cost.

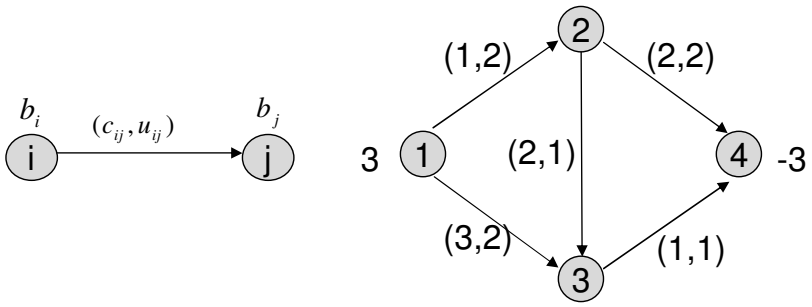

Figure 1.1: An MCF example

The LP formulation for the MCF example in Figure 1.1 is as follows:

$$\min\ x_{12} + 2x_{23} + 2x_{24} + 3x_{13} + x_{34} \tag{1.1}$$

$$s.t. \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ -1 & 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 1 \\ 0 & 0 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{23} \\ x_{24} \\ x_{13} \\ x_{34} \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 0 \\ -3 \end{bmatrix}$$

$$0 \le x_{12} \le 2\ ,\ 0 \le x_{23} \le 1\ ,\ 0 \le x_{24} \le 2$$

$$0 \le x_{13} \le 2\ ,\ 0 \le x_{34} \le 1$$

Since each column in the constraint matrix for an MCF linear programming (LP) formulation contains exactly one 1 and one -1, its basis in fact corresponds to a spanning tree in the network, which makes the simplex method for solving MCF problems firstly introduced by Dantzig (1951) more advantageous since the basis updating procedures can be conducted graphically rather than algebraically. In particular, in a network simplex algorithm, pivoting an entering arc into the basis will create a cycle along which we can increase the flow as much as possible until a leaving arc is identified to be pivoted out. Although the basis updating operations can be efficiently implemented in the network simplex algorithm, the degeneracy issues troubling the original simplex algorithm still remain and make the algorithm cycling without improving the objective value in some intermediate operations. Such cycling operations may even repeat infinitely with some poor pivot rule. Although some anti-cycling rules have been proposed to avoid infinite cycling, degenerate pivots may still take place very often and slow down the algorithm.

The primal-dual algorithm (PD) (Ford and Fulkerson, 1962) is an effective dual-based algorithm but does not move its intermediate solutions along the border of

feasible reason as the dual simplex algorithm does. It was originally developed from a less general algorithm devised for network flow problems, and later was modified for general LPs. In particular, starting from a feasible dual solution, the primal-dual algorithm iteratively identifies columns satisfying the complementary slackness conditions (i.e. columns corresponds to tight dual constraints, or in other words, columns with zero reduced costs) to construct a phase I problem, solves the phase I problem and uses the optimal dual solutions of the phase I problem (if the optimal solution for the phase I problem still contains infeasibility to the original problem) to improve the current feasible dual solution, until the optimal solution for the phase I problem contains zero infeasibility which means the primal feasibility has been attained and thus the optimality for the original problem has been achieved.

Recently a new primal-dual method called the least-squares primal-dual (LSPD) algorithm (Barnes et al., 2002) was proposed to solve LPs. Unlike the original primal-dual algorithm which usually uses simplex-based method (and thus may still be slowed down by degenerate pivoting) to solve the phase I problem, the LSPD algorithm solves a nonnegative least-squares (NNLS) phase I problem firstly proposed by Leichner (1993). By this change, the LSPD algorithm will strictly improve the objective function in each intermediate operation and thus guarantees nondegenerate pivoting at any time.

Although the LSPD algorithm pivots nondegenerately at any time, solving the NNLS subproblem for general LPs may be time-consuming. Nevertheless, the special constraint matrix of the MCF problem makes more efficient NNLS solution methods possible. In particular, the least-squares network flow (LSNF) algorithm proposed by Gopalakrishnan et al. (2004) is a modified LSPD algorithm designed specifically for solving the MCF problem. It has been reported to have good performance in solving some highly degenerate problems such as the assignment problem. Wang (2004) also

applies the LSPD algorithm for solving the one-to-one and one-to-all shortest paths on graphs with nonnegative arc lengths, and shows the LSPD algorithm in fact performs exactly the same steps as the well-known Dijkstra's algorithm (Dijkstra, 1959) does in each iteration. All of these results demonstrate the potential of the LSPD algorithm.

Although the LSNF algorithm has been applied for solving the MCF problem, its original version by Gopalakrishnan et al. (2004) is still incomplete in the sense that it can only be applied on networks with unlimited arc capacities and nonnegative arc lengths. Furthermore, it may be interesting to see whether specialized LSPD algorithms for solving the network optimization problems such as the maximum flow or minimum cost flow problem can be derived to have better efficiency. These are the issues going to be investigated in this thesis.

1.2 Objective and proposed approaches

In this paper, we study the techniques integrating the nonnegative least squares method with the primal-dual algorithms for solving network optimization problems. First, we propose techniques to deal with capacitated networks and compute initial feasible dual solutions for network with some negative arc lengths but not negative cycles, which makes our method more general than the LSNF algorithm proposed by Gopalakrishnan (2004). We then reverse the role of primal and dual formulation and propose a least squares dual-primal (LSDP) algorithm to solve the max-flow problem. In each dual-primal iteration, the NNLS subproblem can be treated as solving for the current on an electronic network with diode and thus Kirchhoff's laws on both voltage and currency can be applied. Insights about this equivalence will be discussed. We also study the use of LSDP algorithm in solving the minimum cost flow problem.

1.3 Scope and limitation

This thesis focuses on efficient LSPD algorithms for different types of network flow problem such as the capacitated MCF problem, and the maximum flow problem. Our methodology is based on the following assumptions and limitations:

1. We only consider directed graphs (i.e. each arc has an orientation from its tail node to its head node).

2. We assume no self-loop (i.e. an arc with the same end nodes) and parallel arcs (i.e. arcs with the same end nodes).

3. The flow in each arc has a zero lower bound and a positive integral upper bound (i.e. capacity). When we say an arc has unlimited capacity, we mean its capacity is a very large integer so that the arc will never be saturated in any case.

4. All the numbers associated with nodes and arcs are deterministic. That is, we only consider the case where the numbers are given beforehand and will not be changed at all times. Also, the network topology is fixed as given.

5. We assume the length for each arc to be an integer (either positive or negative) but there is no negative cycle (i.e. a cycle with total length to be negative).

1.4 Overview of thesis

The rest of this thesis is organized as follows. Chapter 2 reviews the MCF and max-flow problem, as well as their solution methods including the network simplex algorithm, primal-dual algorithm, least-squares primal-dual algorithm (LSPD) (Barnes et al., 2002), and least-squares network flow algorithm (LSNF) (Gopalakrishnan et al., 2004) in the literature. In Chapter 3, we modify the LSNF algorithm for solving the

MCF problem on capacitated networks with negative arc lengths. Our new LSDP algorithm which exchanges the role of the primal and dual formulations in the LSPD algorithm and applies the Kirchhoff's laws to solve an NNLS subproblem will be used to solve the max-flow and MCF problems in Chapter 5. Chapter 6 concludes the thesis and give suggestions for future research.

CHAPTER II

LITERATURE REVIEW

In this Chapter, we first introduce the MCF problem together with its solution property in Section 2.1, then we review the conventional simplex based method for solving the MCF problem as well as its degeneracy issues in Section 2.2. The new least-squares related methods such as the nonnegative least-squares primal-dual (LSPD) algorithm and least-squares network flow (LSNF) algorithms will be discussed in Section 2.3. Finally we will also introduce the max-flow problem and review its solution methods in Section 2.4 and Section 2.5.

2.1 Minimum cost network flow problem

The MCF problem is one of the major network optimization problems discussed in the literature. With proper transformation, some popular network optimization problems such as the shortest path problem and the maximum flow problem can both be viewed as the special cases of the MCF problem. Understanding its problem and solution structures will help researchers to develop more efficient solution methods than general LP solution methods.

2.1.1 Notation and formulations

Let $G = (N, A)$ be a directed graph, where N is the node set with $|N| = n$, and A is the arc set with $|A| = m$. Each arc (i, j) is associated with flow bounds l_{ij} (lower bound) and u_{ij} (upper bound), as well as a unit flow cost c_{ij} (arc length). Without loss of generality, we assume $l_{ij} = 0$ in this thesis. Each node i is associated with an integer b_i to denote the amount of its supply ($b_i > 0$) or demand ($b_i < 0$). A node i is either an S-node (source node) with $b_i > 0$, a T-node (terminal node) with $b_i < 0$, or an O-node (transshipment node) with $b_i = 0$. Therefore, we let $N = S \cup T \cup O$, where $S = \{i : b_i > 0\}$, $T = \{i : b_i < 0\}$, and $O = \{i : b_i = 0\}$. An MCF problem can be formulated as an LP with decision variables x_{ij} representing flow for each arc $(i, j) \in A$. The LP formulation for a general MCF problem is as follows:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.1}$$

$$s.t. \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = b_i \begin{cases} > 0 & \forall\ i \in S \\ = 0 & \forall\ i \in O \\ < 0 & \forall\ i \in T \end{cases}$$

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

When converting a shortest path problem into an MCF problem, we do not consider the arc capacity, thus the u_{ij} is set to be ∞. A one-to-one shortest path problem can be formulated as an MCF problem by assigning the source node one unit of supply and the terminal node one unit of demand. Likewise, a one-to-all shortest path problem can also be viewed as sending $n - 1$ units of supply to the other $n - 1$ demand nodes where each demand node requests one unit of demand.

A conventional $s-t$ maximum flow problem which seeks the max-flow from node

s to node t on a capacitated network can be formulated as follows:

$$\max\ v \tag{2.2}$$

$$s.t. \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} v & \text{if } i = s \\ 0 & \forall\ i \in O \\ -v & \text{if } i = t \end{cases}$$

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

Since to maximize v equals to minimize $-v$, we may convert an $s-t$ max-flow problem into an MCF problem by treating v as the flow along an uncapacitated artificial arc (t,s) with unit cost $c_{ts} = -1$:

$$\min\ -x_{ts} \tag{2.3}$$

$$s.t. \sum_{(i,j)\in A\cup(t,s)} x_{ij} - \sum_{(j,i)\in A\cup(t,s)} x_{ji} = 0$$

$$0 \le x_{ij} \le u_{ij} \quad \forall (i,j) \in A \cup (t,s), \text{ where } u_{ts} = \infty$$

Besides the shortest path problem and maximum flow problem, there are other fundamental network optimization problems such as the transportation problem and the assignment problem which can also be converted into the MCF problem. Since these fundamental network optimization problems are so much related, an efficient MCF solution method should have good potential to be specialized as an efficient solution method for other related fundamental network optimization problems.

2.1.2 Solution properties

Due to the special structure of the MCF problem, we list some mathematical properties the MCF solution should possess:

1. Feasibility condition:

   Besides the arc flow should satisfy their bound constraints, an MCF problem

has a feasible solution if and only if $\sum_{i=1}^{n} b_i = 0$. That is, the sum of the outflow of S-nodes should be equal to the sum of the inflow of T-nodes. We can add a dummy node to take off the extra supply/demand if the condition is not satisfied.

2. Integrality property for each basic feasible solution:

   If b_i is an integer for each $i \in N$ and u_{ij} is a nonnegative integer for each $(i, j) \in A$, each basic solution of the MCF problem will be integers.

3. Spanning tree property for each basic feasible solution:

   Due to the feasibility assumption, we know $\sum_{i=1}^{n} b_i = 0$. Also, since each column in the constraint matrix of an MCF problem respresents an arc which contains two nonzero coefficients, $+1$ for the tail node and -1 for the head node, the flow balance constraints containing n rows will have one redundant row. In other words, every basic solution contains $n - 1$ basic variables (arcs) which are independent to each other. Thus, these basic variables form a spanning tree of the network.

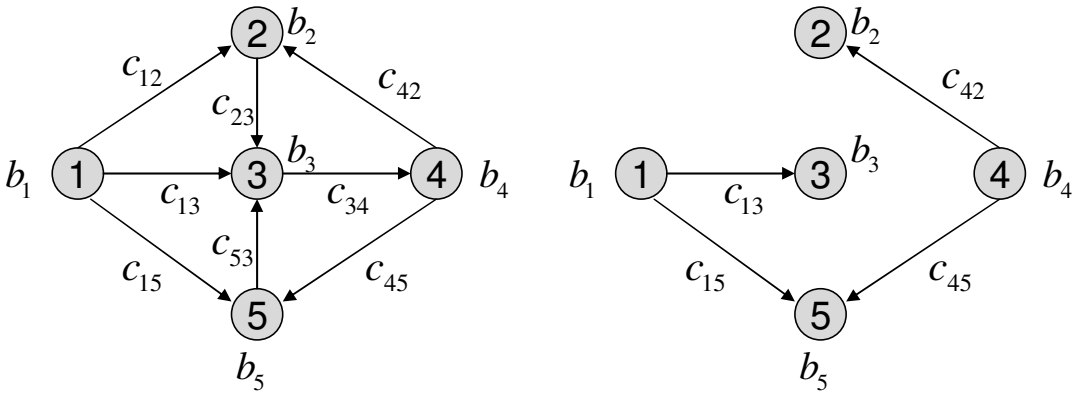

Figure 2.1: An MCF and its basic spanning tree

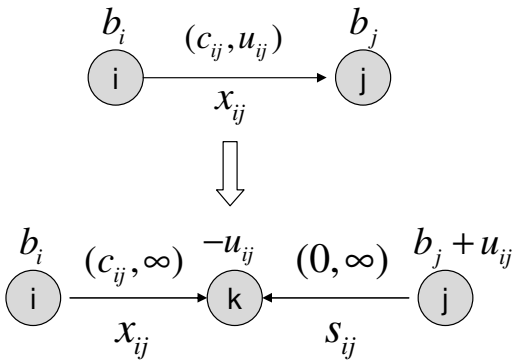

Figure 2.2: Network transformation

2.1.3 Capacitated and uncapacitated MCF

According to the capacity constraints we can set MCF problems into two sets, capacitated and uncapacitated MCF problems. The uncapacitated MCF problem implies no capacity constraint or infinite capacity for each arc. In this section we also discuss capacitated MCF problems where each arc is associated with a capacity constraint. For the simplex method (Dantzig, 1948), we set constraints into two parts, the ordinary constraints and the nonnegative constraints. How we treat the capacity constraints would lead to different situations.

For a capacitated MCF problem, if we take the capacity constraints $x_{ij} \leq u_{ij}$ for each $(i, j) \in A$ as parts of the ordinary constraints, the amount of the ordinary constraints will increase from n to $n + m$. And the number of variables will also increase from m to $2m$ after we turn the constraints into standard form $x_{ij} + s_{ij} = u_{ij}$ by adding the slack variables s_{ij} to each capacity constraint. After introducing the these slack variables, we can convert a capacitated problem into an uncapacitated problem. Figure 2.2 shows the graph transformation which turns a capacitated arc into two uncapacitated arcs.

On the other hand, we can view the capacity constraints as parts of the nonnegative constraints which are used to calculate the minimum ratio to determine the

pivoted out arc at each iteration. A basic variable will be pivoted out when it decreases to its lower bound (i.e.zero in this thesis). Similarly, we also pivot a basic variable x_{ij} out when it increases to its upper bound (i.e. (i, j) is saturated), that implies the slack variable s_{ij} of the capacity constraint decreases to zero. Therefore, in that situation we can replace the decision variable x_{ij} by its slack variable s_{ij} which is at its lower bound, and pivot s_{ij} out. In the case of an MCF problem, when arc (i, j) is saturated, we replace arc (i, j) by a reversal arc (j, i) and let x_{ji} which suggests the slack variable of x_{ij} denote the flow on it, and then arc (j, i) will be pivoted out.

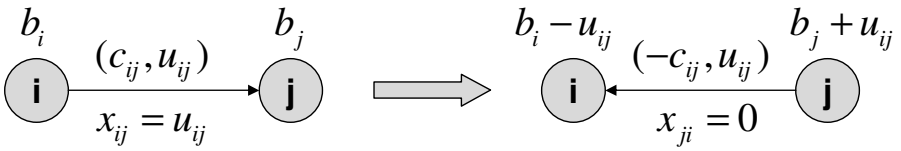

Figure 2.3: Arc reversal transformation

2.2 Review of the solution methods for solving the MCF problem

The simplex algorithm is an effective method for solving LPs. Since the MCF problem is a specialized LP, the simplex algorithm is applicable for solving the MCF problem. In this section, we first introduce the the graphical simplex algorithm designed for solving network flow problems called as the "network simplex algorithm" (Dantzig, 1951), and then we introduce the generic primal-dual algorithm for the MCF problem.

2.2.1 Network simplex algorithm and the degeneracy issues

Network simplex algorithm is a well-known method for solving the MCF problem. The graphical algorithm represents the procedure corresponding to each step of the simplex algorithm, such as determining the entering variable, calculating the minimum

ratio to identify the leaving variable, and updating the dual variables associated with each node.

For the node-arc incident formulation, each constraint represents a node flow balance, and variables denote the flow upon arcs. We start with a basis which is a spanning tree, and determine the entering variable by calculating the dual slack variables, $c_{ij} - p_i + p_j$. Since a basic solution corresponds to a spanning tree, we calculate the dual variable (each dual variable is associated with a node, therefore we may call a dual variable as a node potential) for each node according to constraint $c_{ij} - p_i + p_j = 0$ of each basic arc. After obtaining the node potentials, we can decide the entering arc by examining the value of $c_{ij} - p_i + p_j$ for each nonbasic arc. According to Figure 2.1 we know that as an additional arc enters the basis, the original $n - 1$ basic arcs plus the additional one will form a cycle, and when the entering arc increases the flow from zero, the flow of some arcs in the cycle backward to the entering arc will decrease. Therefore, we can pivot out the arc whose flow drops to zero first, and that is equal to the process of calculating the minimum ratio.

Although the simplex algorithm is an effective method for solving LPs, the degenerate pivoting may slow down the algorithm. Degeneracy is the situation that more than n constraints are tight at one point in R^n. Therefore, the degenerate pivoting suggests that we have more than one choice when we are selecting the pivot-out variable and we may be stuck at one point (with different basis, since there are more than n constraints tight at the point) without any improvement for a period due to the wrong choice of leaving variable. In particular, if there are k constraints tight at the point, $k > n$, there will be C_n^k basis representing the same point

When we select an entering arc to create a cycle, we have to determine the leaving arc by identifying the arc with flow drops to zero (or increases to the upper

bound) while the flow of the entering arc increases. When there are more than one arc satisfying the pivot out condition simultaneously, that is so-called degenerate pivoting. And when there is a basic arc (i, j) whose flow is zero, it may lead to that an entering arc backward to arc (i, j) can not increase any flow since there is no flow on (i, j) can be reduced. Therefore how to decide which one to leave the basis would be critical to the rest of the operations. The worst case for degenerate pivoting is cycling, an infinite repetitive sequence of degenerate pivots. In that case, the simplex algorithm would not terminate in a finite period.

There are different pivot rules for the network simplex algorithm (Bazaraa, Jarvis, and Sherali, 1990), such as "Dantzig's pivot rule","first eligible arc pivot rule", "candidate list pivot rule", and so on. Different pivot rules have different drawbacks and are suitable for different situations. Dantzig's pivot rule selects the arc with steepest ascent/descent improvement as the entering arc.

When we are facing a degenerate problem, we may stay at a non-optimal extreme point and pivot through a sequence of associated basis. If the same sequence of pivots is used over and over again, we will cycle forever among the sequence of basis, and we repeat iterating the basis without reaching an optimal condition. Poor pivot rule will lead to cycling. There are different pivot rules proposed for anti-cycling, they guarantee finite convergence of the simplex algorithm by ensuring that no basis can be repeated, such as "Lexicographic rule" (Bazaraa et al., 1990) and "Bland's rule" (Bazaraa et al., 1990). "Leaving arc rule" (Cunningham, 1976) is also a cycling prevention rule for network flow problem, that keeps strongly feasible spanning trees as Figure 2.4 at every iteration. Recently, many researchers are trying to propose cycling prevention method with improved efficiency, Sokkalingam and Ahuja (Ahuja et al., 2002) proposed new pivot selection rules for the network simplex algorithm.

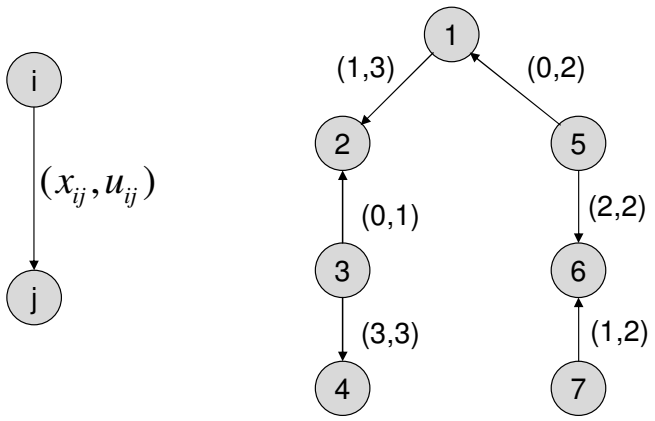

Figure 2.4: A strongly feasible tree

2.2.2 Generic primal-dual algorithm

Besides the simplex algorithm, there are other well-known methods for solving LPs, the primal-dual algorithm (Ford and Fulkerson, 1962) is one of them. The primal-dual algorithm was originally proposed for network flow problem and then modified for solving general LPs. It has also been studied for years, different versions have be proposed for better efficiency, such as "steepest-edge primal-dual simplex method "(Curet, 1998), "revised primal-dual algorithm "(Paparrizos et al., 2003) and "least-squares primal-dual algorithm "(Barnes et al., 2002). Compared with the simplex method that proceeds by keeping primal feasible until the complementary dual solution is feasible (the optimal condition), the primal-dual algorithm is similar to the dual simplex method which begins with a dual feasible solution. An important property of the primal-dual algorithm is that the dual solution is not necessarily basic. According to the complementary slackness property, we have a primal basis corresponding to those tight dual constraints. If the primal basis is not feasible we try to reduce the primal infeasibility. At every iteration, we find an improving direction rather than the pivot operation of the simplex method to move to the next solution. In other words, different from the simplex algorithm that always moves along the boundaries

of the feasible region so that each intermediate solution must be a corner point, the primal-dual algorithm obtains the intermediate solution by moving along the improving direction through the feasible region, and that is why the solution of the primal-dual algorithm is not necessarily basic.

Consider the following problem in standard form:

$$\begin{aligned} &\min\ cx \\ &s.t.\ \ Ax = b \\ &\qquad x \geq 0 \end{aligned} \tag{2.4}$$

c is a $1 \times m$ vector, b and the decision variable x are $n \times 1$ and $m \times 1$ vectors respectively, and A is the $n \times m$ constraints matrix. We form the corresponding dual problem as:

$$\begin{aligned} &\max\ pb \\ &s.t.\ \ pA \leq c \end{aligned} \tag{2.5}$$

p is a $1 \times n$ vector denoting the dual variables. We let A_i be the ith column of matrix A and c_i stands for the ith element of c vector. According to the complementary slackness property, if the ith constraint is tight, $pA_i = c_i$, then x_i is a primal basic variable. So that base on the dual solution we can check if this corresponding basis is primal feasible or not. By letting $Q = \{i : pA_i = c_i\}$, we can form the restricted primal problem (RP) as follows.

$$\begin{aligned} &\min\ \sum_{j=1}^{n} |s_j| \\ &s.t.\ Ax + Is = b \\ &\qquad x_i \geq 0 \ \ \forall\ i \in Q \\ &\qquad x_i = 0 \ \ \forall\ i \notin Q \end{aligned} \tag{2.6}$$

RP is a phase one problem, s is an $n \times 1$ vector denotes the artificial variables (the infeasibility), s_i means the ith element of s. While the objective value $\sum_{j=1}^{n} |s_j|$ is minimized to zero, these basic variables can construct a primal feasible solution which is the optimal condition of the dual problem. At every iteration we move to a better dual intermediate solution that admits more variables into the primal basis to decrease the objective value of RP. We modify a dual vector v as an improving direction of p, and let $p' = p + \theta v$ be a new dual solution for the next iteration. p' has to introduce at least one new variable into the primal basis to reduce the infeasibility. The conventional primal-dual algorithm acquires the improving direction by solving the dual problem of RP, so-called DRP, as follows:

$$\begin{aligned} &\max\ vb \\ &s.t\ \ vA_i \leq 0 \quad \forall\ i \in Q \\ &\qquad v \leq 1 \end{aligned} \tag{2.7}$$

Before we come to the DRP, one more point of improving direction must be clarified. In other words, improving direction must be able to improve the objective value and it must also be a feasible direction. Therefore, according to (2.5), we have $p'b = (p + \theta v)b \geq pb$, and $p'A = (p + \theta v)A \leq c$. So that for every improving direction it must satisfy the condition that $vb \geq 0$, and $vA_i \leq 0$ for $i \in Q$ since $pA_i = c$.

Now we let v^* denote the optimal solution of DRP, it is qualified as an improving direction since it satisfies the condition that $v^*b \geq 0$ and $v^*A_i \leq 0$ for $i \in Q$. After obtaining the improving direction, we move along the direction v^* till we are bound by a constraint i such that $v^*A_i > 0$ for $i \notin Q$. Therefore, we have to find out which constraint is binding and how much we can improve when we move along the improving

direction. We define the step length θ as follows:

$$\theta = \frac{-(pA_k - c_k)}{v^* A_k} = \min \left\{ \frac{-(pA_i - c_i)}{v^* A_i} : v^* A_i > 0 \right\} > 0 \tag{2.8}$$

After we have the improving direction v^* and the improving step length θ, we can modify a new dual solution p' for the next iteration.

Since we have to acquire v and θ before we move to the next intermediate solution, if there is no i such that $v^* A_i > 0$ for $i \notin Q$, we can move along the direction without reaching any boundary, which means the dual problem is unbounded and the primal problem is infeasible.

DRP is also an LP which is usually solved by the simplex related method and thus it may perform degenerate pivoting. Now, we will introduce a new revised primal-dual method that can exterminate that degenerate pivoting by replacing the DRP with a nonnegative least-squares (NNLS) problem to acquire the improving direction.

2.3 Review of the least-squares related solution methods for MCF

Least-squares problems minimize a quadratic objective problem which is restricted to the linear constraints. Leichner (1993) has proposed the algorithm for solving the phase I problem by formulating the phase I problem as a least-squares problem. Since the RP of the primal-dual method is also a phase I problem, Barnes (2002) proposed the least-squares primal-dual (LSPD) algorithm which combines the primal-dual algorithm and least-squares problem to solve LPs with good efficiency, he also proved that the LSPD algorithm has the steepest ascent/descent improving direction. Recently, Gopalakrishnan (2004) noticed that the least-squares property of the network flow problem due to its constraint structure is advantageous on solving a least-squares problem, he established the least-squares network flow algorithm (LSNF)

by utilizing the advantages to solve MCF problems effectively.

2.3.1 Least-squares primal-dual algorithm

The following is a nonnegative least-squares (NNLS) problem,

$$\min \ ||b - Ax||^2 \tag{2.9}$$

$$s.t. \ x \geq 0$$

where b and x are $n \times 1$ and $m \times 1$ vectors respectively, A is an $n \times m$ matrix. We can describe the formulation in geometrical aspect that we want to combine the column vectors in matrix A through a nonnegative weight x to minimize the distance between Ax and b.

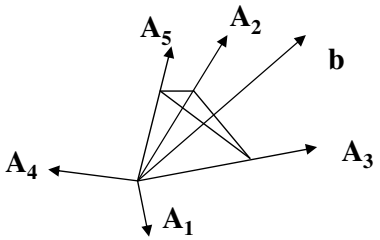

Figure 2.5: Graphical representation of the infeasibility and the NNLS problem

Since the objective is to make the distance between b and Ax as short as possible, if b is in the cone formed by A as Figure 2.5, then the objective is zero. Else, we let Ax be the projection of vector b on the cone, then $b - Ax$, the distance between b and cone A, will be minimized. We let $s = b - Ax$ and rewrite (2.9) as:

$$\min \ ||s||^2 \tag{2.10}$$

$$s.t. \ Ax + Is = b$$

$$x \geq 0$$

From the geometric aspect as shown in Figure 2.6, we know that $sb > 0$ and $sA_i \leq 0$. Therefore, s can be qualified as an improving direction.

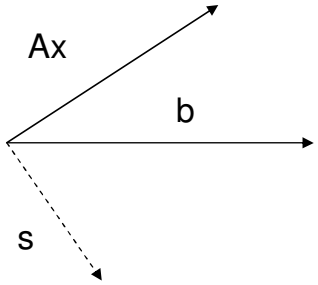

Figure 2.6: The infeasibility

2.3.2 Least-squares network flow algorithm

After the LSPD (Barnes et al., 2002) algorithm was proved to be the steepest descent/ascent primal-dual method, Gopalakrishnan (2004) applied the concept of LSPD on solving the uncapacitated MCF problem by utilizing the least-squares property of the network flow problem. Furthermore, Wang (2004) has also proved that the procedure of the LSPD algorithm on solving shortest path problems with nonnegative arc lengths is equal to the Dijkstra's algorithm (Dijkstra, 1959) which is a famous method for solving the nonnegative shortest path problem. The LSNF algorithm proposed by Gopalakrishnan (2004) is a graphical algorithm that each iteration of the process can be demonstrated graphically. Here we describe the least-squares property of network flow problem.

The following is a least-squares problem

$$\min \sum_{j=1}^{n} s_j^2 \tag{2.11}$$

$$s.t.\ Ax + Is = b$$

As the problem has the network flow structure that each variable has two nonzero coefficients 1 and -1, for simplicity, we illustrate the condition of the problem as

follows:

$$\min \sum_{j=1}^{2} s_j^2 \tag{2.12}$$

$$s.t.\ x_1 + s_1 = b_1$$

$$-x_1 + s_2 = b_2$$

The constraints of (2.12) suggest the fact that $s_1 + s_2 = b_1 + b_2$. Since the objective is to minimize $s_1^2 + s_2^2$, we have $s_i^* = s_j^* = \frac{b_i+b_j}{2}$ by letting $x_1^* = \frac{b_i-b_j}{2}$ which is the optimal solution. Therefore, for a least-squares problem with network structure, we can optimize the problem by averaging the infeasibility. The LSNF algorithm first takes the advantage of the least-squares property of network problems to solve the least-squares problem, and then adjust the solution to be nonnegative by the method as the inner loop of the phase I algorithm (Leichner et al., 1993). For these capacitated problems, the LSNF algorithm will turn them into the uncapacitated problems by the arc transformation technique as shown in Figure 2.2.

We can make some analogy for the procedure of the LSNF algorithm before we go further. According to the chemical function "pv=nRt" we know that the pressure has the direct proportion to n, amount of air, and t, temperature. We relate n to the infeasibility of each node. So that at each increase of temperature, the pressure inside each node will increase according to the proportion of the infeasibility associated to each node. By letting the pressure inside node i be $s_i t$ which is δp_i, and the resistance stopping air from leaking out through arc (i, j) be $c_{ij} - p_i + p_j$. we describe the procedure of the LSNF algorithm as follows. We increase the temperature sequentially, each time when an arc is unobstructed those connected nodes will share their the infeasibility averagely. And while whole of the infeasibility of the network are eliminated, that implies the objective value of the NNLS problem is zero and the basis is primal feasible,

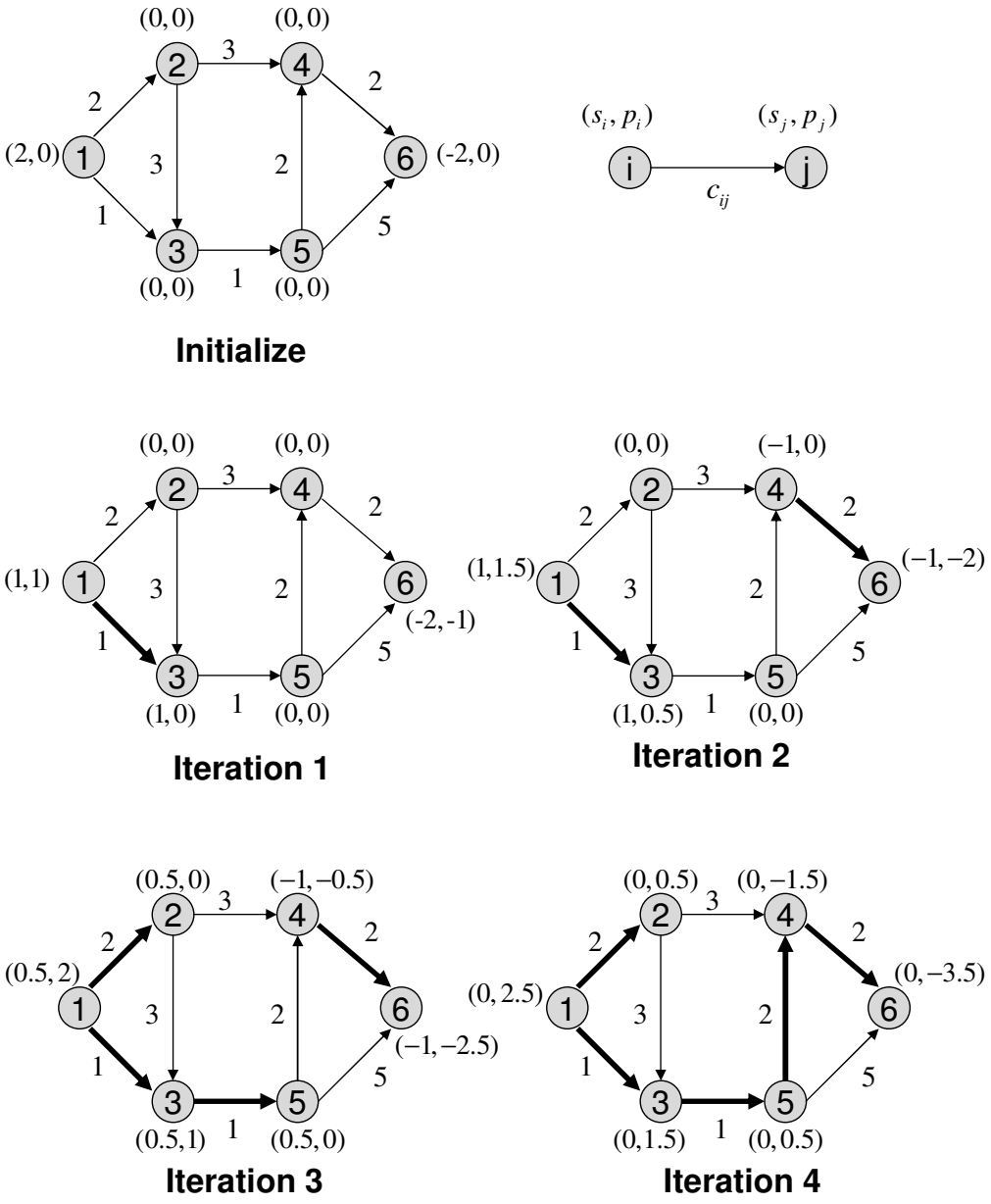

Figure 2.7: An MCF example solved by LSNF

Algorithm 1 The LSNF algorithm

Input a network with node balances b and edge set A

Initialize

pseudo-flow $x := 0$ and node potential $p := 0$;

$E = (i, j) \in A : c_{ij} - p_i + p_j = 0$

While $b \neq 0$ **Do**

if $\exists$ trees T_i and T_j such that $i \in T_i$, $j \in T_j$ and $(i, j) \in A$ with tree imbalances $s_i = \frac{\sum_{k \in T_i} b_k}{\sum_{k \in T_i} 1} > s_j = \frac{\sum_{k \in T_j} b_k}{\sum_{k \in T_j} 1}$ **Then**

$T'_i = T_i \bigcup T_j \bigcup (i, j)$ and tree imbalance $s_{T'} = \frac{\sum_{k \in T'} b_k}{\sum_{k \in T'} 1}$

temporary node balances $b'_k = b_k - s_{T'}$ for all nodes $k \in T'$

the unique partial pseudo-flow $x_{T'} \geq 0$ on tree T' satisfies all temporary node balance b';

$x := x_{T'}$; $s := s_{T'}$

$t := \min_{(i.j) \in E : s_i - s_j > 0} \frac{c_{ij} - p_i + p_j}{s_i - s_j}$;

$p := p + ts$;

Else

stop, output problem is infeasible;

End;

End While;

stop, output x as the optimal flow;

the optimal condition is reached.

We briefly illustrate the LSNF algorithm with Figure 2.7 which is an uncapacitated MCF problem without negative arc lengths. Initially, the pressure in the nodes are zeros, and we put 2 and -2 units in the source node and the terminal node, respectively. The negative unit leads to the force of suck compare to the force of pressure caused by the positive unit. When we start to increase the temperature, the pressure in node 1 increases from 0 to 1, the resistance on arc $(1, 3)$ is equal to the pressure so that arc $(1, 3)$ is unobstructed, then the air in node 1 leaks out to node 3. Therefore, at the end of iteration 1, we have air distributed in node 1 to 6 are $(1, 0, 1, 0, 0, -2)$ respectively. Then we keep increasing the temperature. Likewise, as long as the pressure between node 4 and 6 is equal to the resistance of arc $(4, 6)$, there will be 1 unit of air leaking out from node 4 to node 6 through arc $(4, 6)$. The similar process will be repeated at the following iterations until those unobstructed arcs form an s-t path that

connects the source node and the terminal node, so that the infeasibility is eliminated, the gaol of optimality is achieved.

2.4 Maximum flow problem

The maximum flow problem searches the paths upon a capacitated network to maximize the flow from the source node to the terminal node. According to (2.2) and (2.3) we know that a max-flow problem can be converted into an MCF problem with zero arc lengths except arc (t, s). Since each unit flow of arc (t, s) will reduce 1 objective value, therefore the more the flow on arc (t, s) the better the objective of minimum cost will be. The idea is first proposed by Fulkerson and Dantzig (1955).

The dual problem of the max-flow problem is finding the min-cut. A cut is an arc set $\{(i, j) : i \in X, j \in \bar{X}\}$ which X is a subset of N and $\bar{X} = N - X$. So that when we find an $s - t$ cut with minimum capacity we find the maximum flow. A max-flow problem can have multiple solutions based on one min-cut. Therefore, comparing to general MCF problems, max-flow problems often have multiple solutions due to different flow compositions.

2.5 Review of the solution methods for max-flow problem

Since the max-flow problem can be viewed as a special case of the MCF problem, it also can be solved by those solution methods for MCF. Besides those methods for MCF, there are some specialized algorithm proposed to solve the max-flow problem for better efficiency. An intuitive method, the augmenting path algorithm proposed by Ford and Fulkerson (1956), searches for an augmenting path upon an residual network, and push flow until there is no $s-t$ path can be found. Later, Edmonds and Karp (1972) suggested an implementation of the augmenting path algorithm with better efficiency. The preflow-push algorithm proposed by Karzanov (1974) features the violation of flow

balance constraints at intermediate stages, the algorithm was later revised by Goldberg and Tarjan (1988).

The primal-dual algorithm is also applicable for solving the max-flow problem. The characteristic of the primal-dual algorithm applied on max-flow problem is that we write our problem as the dual stage comparing to the conventional method that the main problem is always taken as the primal problem. So that, we would rather call the method as dual-primal to differentiate from that used for MCF problems. If we take the max-flow problem as the dual problem, we can write the corresponding RP problem according to (2.2) as follows:

$$\begin{aligned} & \min \quad \sum_{(i,j)\in A} r_{ij}u_{ij} \qquad (2.13) \\ s.t. \quad & p_i - p_j + r_{ij} \geq 0 \\ & p_t - p_s \geq 1 \\ & r_{ij} \geq 0 \quad \forall (i,j) \in U \\ & U = \{(i,j) : (i,j) \in A \ ; \ x_{ij} = u_{ij}\} \end{aligned}$$

r_{ij} denotes the dual variable of the capacity constraint of arc (i, j), the second constraint corresponds to variable v. The formulation demonstrates the "max-flow, min-cut theory", where the problem seeks the minimum cut. Therefore, the optimal value of the dual problem, min-cut, is equal to the optimal objective value of the primal problem, the max-flow. According to (2.13), we formulate the DRP problem as follows:

$$
\begin{aligned}
\max \quad & v && (2.14)\\
s.t. \quad & Ax + dv = 0 \\
& x_{ij} \leq 0 \quad \forall (i,j) \in U \\
& x_{ij} \geq 0 \quad \forall (i,j) \in L \\
& x_{ij} \leq 1 \\
& v \leq 1 \\
& L = \{(i,j) : (i,j) \in A \ ; \ x_{ij} = 0\}
\end{aligned}
$$

The problem can be interpreted as a reachability problem in finding a path from s to t by using the following method: arcs with zero flow in the forward direction, saturated arcs in the backward direction, and others in either direction. As we find an $s-t$ path, we push flow along the path, which implies we improve the objective value with the improving direction until an arc reaches the upper bound or lower bound, which corresponds to the operations of the augmenting path algorithm. In fact, the augmenting path algorithm is an application of the primal-dual algorithm.

2.6 Summary

This Chapter contains two major topics, the MCF problem and the max-flow problem. We first introduced notation, solution properties and mathematical formulations for the MCF problem in Section 2.1. Then the literature reviews including the network simplex algorithm, the primal-dual algorithm, the least-squares primal-dual algorithm, and the least-squares network flow algorithm were made in Section 2.2 and 2.3. Literatures related to the max-flow problem is capsuled in Section 2.4 and the following Section 2.5 illustrated the idea of the dual-primal algorithm.

CHAPTER III

SOLVING THE MCF WITH NEGATIVE ARC LENGTH AND CAPACITY CONSTRAINTS

In this chapter, we focus on the methods that can make the LSNF algorithm more comprehensive for solving MCF problems with arc capacity constraints and negative arc lengths. The illustration of our revised LSNF algorithm for dealing with capacitated MCF problems will be given in Section 3.1. How to obtain an initial dual feasible solution for an MCF problem with negative arc lengths will be discussed in Section 3.2. Section 3.3 introduces the application of our revised LSNF algorithm on max-flow problems.

3.1 LSNF for capacitated MCF

Since the uncapacitated MCF problem can be viewed as a special MCF problem with infinite capacity, we will discuss about the application of the LSNF algorithm on capacitated MCF problems to generalize the effect of the LSNF algorithm.

The LSNF algorithm proposed by Gopalakrishnan (2004) is applied on solving uncapacitated MCF problems. Gopalakrishnan (2004) suggests converting a capacitated problem into an uncapacitated one by a technique introduced in Section 2.1.3. There are some disadvantages of this network transformation technique: First, the size of the constraints matrix will increase, while we replace a capacitated arc by two

uncapacitated arcs and an additional node. Second, we have to preprocess all of the capacitated arcs before we solve the problem by the LSNF algorithm, even if some of the arcs are not the bottlenecks for us to achieve the optimum. Therefore, rather than the network transformation technique, we would prefer to solve the capacitated MCF problem with the arc reversal transformation method. In particular, when x_{ij} reaches the upper bound, we replace it by a reverse arc (j, i), and introduce a brand new variable $y_{ji} = 0$, denote the flow on the reverse arc (j, i).

The least-squares property of the network flow problem is to divide the infeasibility equally. With the nonnegative constraints, we solve the nonnegative least-squares (NNLS) problem to acquire the improving direction of the LSNF algorithm. As the additional capacity constraints are taken in consideration, we replace the NNLS problem with the bound least-squares (BLS) problem. By performing the revised LSNF algorithm (solving BLS problems), it is easy to find that the slack vector of the BLS problem remains valid throughout the LSNF algorithm.

Due to the special structure of network flow problems introduced in Section 2.3.2, we can convert the flow balance constraints into several constraints in a form similar to $s_i + s_j = K$, when s denotes the slack variable and K is a constant. From the algebraic aspect, we know that the smaller the difference between s_i and s_j is, the smaller the $s_i^2 + s_j^2$ will be. Therefore, we divide the infeasibility as equal as possible under the capacity and nonnegative constraints. Figure 3.1 shows an example of a BLS problem. In the beginning, there are 4 units and 0 unit of infeasibility associated with node i and j respectively. Since arc (i, j) has 1-unit upper bound, we divide the infeasibility as 3 and 1 by flowing 1 unit through arc (i, j), and then reverse the arc. Later, when we improve the node potentials, arc (j, i) will leave the basis.

Now we illustrate how the LSNF algorithm (Gopalakrishnan et al., 2004) works

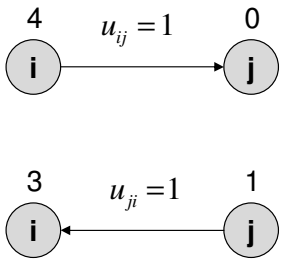

Figure 3.1: A small BLS example

on solving a capacitated problem similar to the example in Figure 2.7. We assume $u_{35} = 1$, thus $x_{35} \leq 1$, and the process will be different from Figure 2.7 after step 4. Figure 3.2 shows the rest of steps since arc $(3, 5)$ becomes a basic arc. As arc $(3, 5)$ enters the basis at step 3, we can calculate $(x_{12}, x_{13}, x_{23}, x_{24}, x_{35}, x_{54}, x_{46}, x_{56})$ to be $(\frac{1}{2}, 1, 0, 0, \frac{1}{2}, 0, 1, 0)$ respectively by $b = (2, 0, 0, 0, 0, -2)$, and the infeasibility s associated with each node. At step 4, the flow on arc $(3, 5)$ is supposed to be 2 units if there was no capacity constraint. Since arc $(3, 5)$ can only allow 1 unit flow, we make use of the arc reversal transformation by letting the flow on arc $(3, 5)$ be 1 unit which is its upper bound ,then reverse arc $(3, 5)$ and let y_{53} denote the flow on the reverse arc. After that, we update $b = (2, 0, 0, 0, 0, -2)$ to $b = (2, 0, -1, 0, 1, -2)$, and we have the solution $(x_{12}, x_{13}, x_{23}, x_{24}, y_{35}, x_{54}, x_{46}, x_{56}) = (\frac{1}{3}, \frac{4}{3}, 0, 0, 0, \frac{4}{3}, \frac{5}{3}, 0)$ according to $b = (2, 0, -1, 0, 1, -2)$ and $s = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{-1}{3}, \frac{-1}{3}, \frac{-1}{3})$. The last step shows arc $(2, 4)$ enters the basis and the optimal solution is composed of 1 unit flow through the path $1 - 3 - 5 - 4 - 6$ and another unit flow through the path $1 - 2 - 4 - 6$ with total cost 13.

3.2 Methods to obtain an initial dual feasible solution

In real-world application, the arc lengths are usually nonnegative for the MCF problem, since there is always a positive cost to transport commodities. However, in some situations (e.g. recycling network) a negative arc length may be treated as a

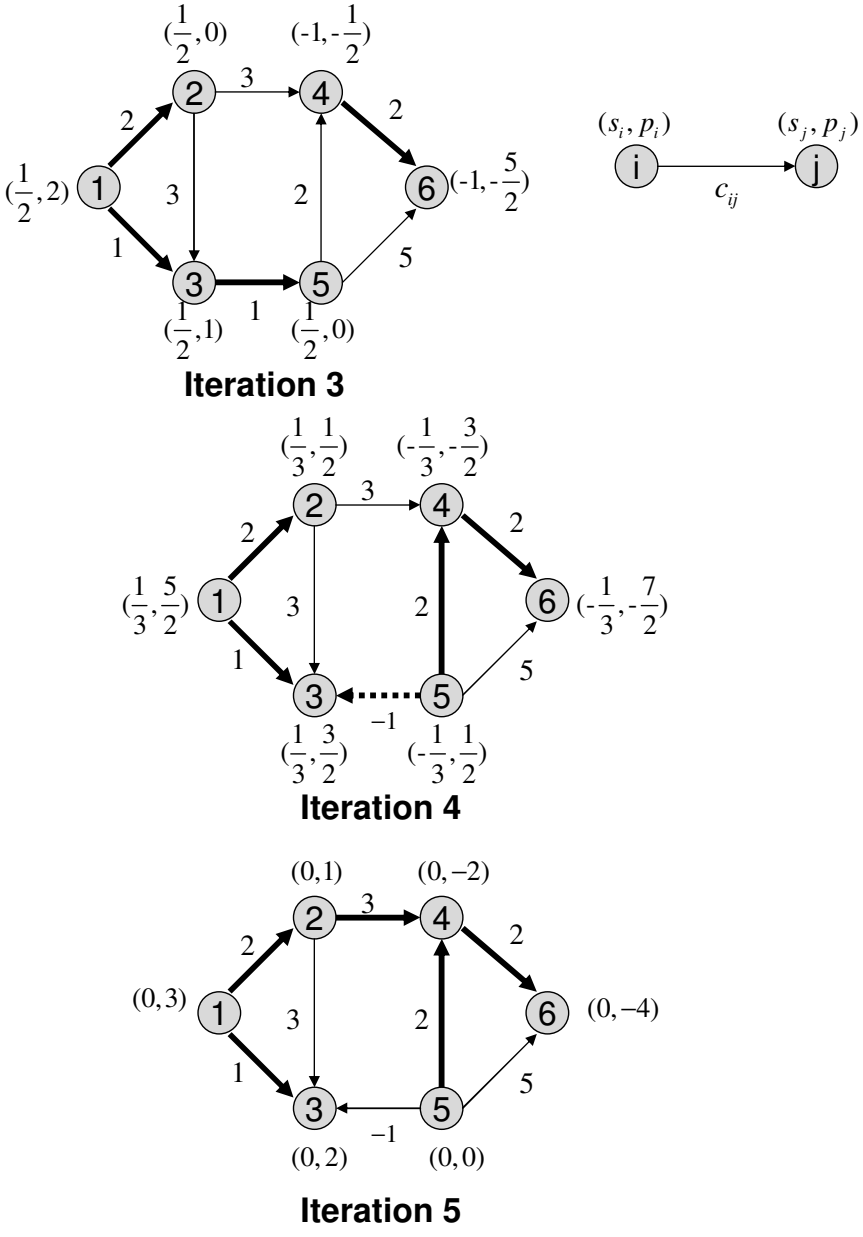

Figure 3.2: LSNF on a capacitated network

benefit, and thus we may deal with an MCF problem with negative arc lengths. For an MCF problem with nonnegative arc lengths, a zero vector is a trivial dual feasible solution. For problems with negative arc lengths, we need some technique to acquire an initial dual feasible solution. Traditionally, when the initial dual feasible solution is not obvious, the artificial constraint technique (see Section 6.6 in (Bazaraa et al., 1990)) is an effective method to get an initial dual solution. It works by adding a single new primal constraint with a large right hand side (RHS) to get a dual feasible solution. In fact, the additional constraint corresponds to an artificial dual variable, and the constraint with a large RHS would never be tight implies the artificial dual variable can not be in the basis.

Although the artificial constraint technique is an effective way to get a dual feasible solution for problems with negative arc lengths, we do not suggest using it for network flow problems. For a node-arc incident matrix, the pair of 1 and -1 in each column forms the special structure that leads to the least-squares property of the network flow problem. As we add an additional constraint, this special structure will be destroyed, and the technique to calculate the optimal infeasibility for each node by averaging the infeasibility at a component can not be applied. Therefore, we have to come up with an effective method that can provide us an initial dual feasible solution without the drawback of the artificial constraint technique.

The arc reversal technique for dealing with a capacitated MCF can also be applied to obtain an initial dual feasible solution. As shown in Figure 2.3, when we reverse arc (i, j), the arc length of the reverse arc (j, i) becomes $-c_{ij}$, and b_i and b_j become $b_i - u_{ij}$ and $b_j + u_{ij}$ respectively. Therefore, we can use the arc reversal technique to transform all of the arcs with negative arc lengths to be nonnegative. However, the arc reversal technique is only applicable for capacitated MCF problems (i.e. u_{ij} has

to be a finite number for each $(i, j) \in A$). To deal with the case where some arc has unlimited capacity, we suggest using the label-correcting like technique.

The label-correcting algorithm (Ford, 1956) is a well-known method for solving the shortest path problem with negative arc lengths. Now, we would like to follow the idea of the label-correcting algorithm to help us get an initial dual feasible solution.

This method maintains a node potential p_i for each node $i \in N$. The objective is to find a dual solution such that $p_j \geq p_i - c_{ij}$ for all $i, j \in N$, and $(i, j) \in A$. First of all, we set $p_i = 0$ for all $i \in N$, and check the dual feasible condition $p_j \geq p_i - c_{ij}$ for each arc $(i, j) \in A$. If we detect an arc such that $p_j < p_i - c_{ij}$, we update p_j to be $p_i - c_{ij}$. The process continues until no more arc violating the dual feasible condition. Thus when the process terminates, the node potentials are dual feasible.

Algorithm 2 Label-correcting like technique

$p_i = 0$ for $i \in N$;
While some arc (i, j) satisfies $p_j < p_i - c_{ij}$ **do**
    $p_j = p_i - c_{ij}$;
End while;

Here we give an algorithm named MLCL algorithm (MLCL stands for modified label-correcting like), which is an efficient implementation of the label-correcting like technique described in previous paragraph. First of all, we assign a zero node potential to each node. We identify those arcs with negative arc lengths since they are violating the dual feasible condition, and let $S = \{j : c_{ij} < 0\}$, $p_j = -c_{ij}$ for all $j \in S$. Set S contains nodes i whose potential are to be altered. We remove nodes from set S, update the potential of their emanating nodes, and then add the nodes whose potential are modified back to S. These steps repeat until S becomes empty. In particular, at each iteration we will take an element i out of set S, while we detect that $p_j < p_i - c_{ij}$ for $(i, j) \in A$ we will update p_j to be $p_i - c_{ij}$ and put j into set S since the node potential p_j is changed. Finally when set S is empty, no arc needs to be verified and

the dual feasible solution is obtained.

Algorithm 3 MLCL algorithm

```
p_i = 0 for i ∈ N;
S = {j : c_ij < 0, (i, j) ∈ A};
p_j = −c_ij
While S ≠ ∅ Do
    remove an element i from S;
    For each arc(i, j) ∈ A Do
        If p_j < p_i − c_ij Then
            p_j = p_i − c_ij;
            If j ∉ S Then
                add node j to S;
End While;
```

Figure 3.3 shows an example of applying the MLCL algorithm. In the beginning, we put node 2 and 4 into set S and update their node potential since the lengths of arc $(1, 2)$ and $(2, 4)$ are both negative. At iteration 1, we take element 2 out of set S, and check the dual feasible condition for its outgoing arc. For arc $(2, 4)$ we have $3 < 2 - (-3) = 5$, so we update the node potential of node 4 from 3 to 5. Likewise, at iteration 2 we update the node potential of node 6 from 0 to 3. Finally, when we take element 6 out of S, S becomes empty and we have a dual feasible solution $(0, 2, 0, 5, 0, 3)$.

The process of this technique is similar to the label-correcting algorithm. In the worse case, we may update all of the node potential to get a dual feasible solution. Thus the complexity of the technique is $O(nm)$, the same as the label-correcting algorithm. However, since setting zero potential for each node may already satisfy the dual feasible condition for most arcs, and MLCL algorithm only propagates on those nodes with updated potential, the MLCL algorithm may not always take as much time as the original label-correcting algorithm requires in practice.

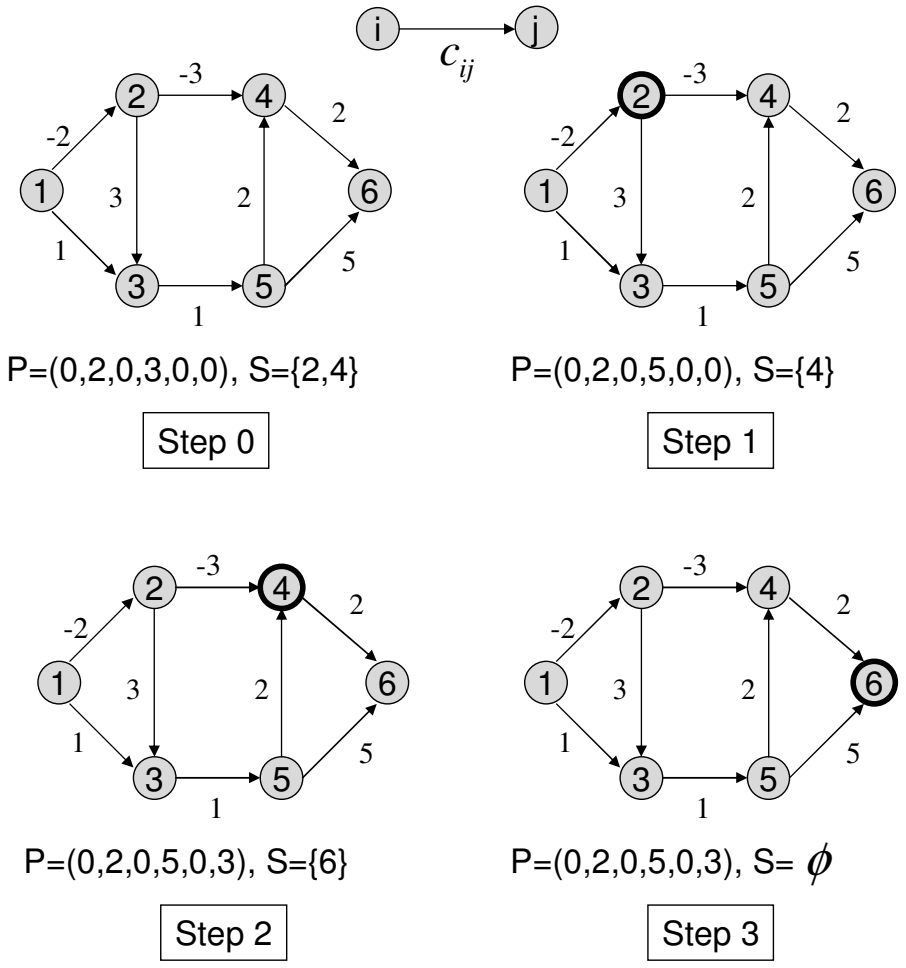

Figure 3.3: Label-correcting like technique

3.3 LSNF for max-flow problem

Since the shortest path problem and the max-flow problem are special cases of the MCF problem, the LSNF algorithm is also applicable to these problems. The application to solve the shortest path problem have been proved to perform the identical steps as the Dijkstra's algorithm by Wang (2004). The idea of the LSNF algorithm is to average the infeasibility as equal as possible, however, for the max-flow problem, the infeasibility is zero in the beginning. Therefore before conducting the a max-flow problem by the LSNF algorithm we should have some preprocesses. First of all, we add an artificial (t, s) arc with -1 arc length to turn the max-flow problem into an MCF problem. Then we adopt the technique of pre-flow push to allocate certain infeasibility as shown in Figure 3.4, the value beside nodes are the infeasibility and those attached to arcs are the capacities. We push flow from the source node to its emanating nodes to saturate the outgoing arcs and reverse them, then we let the node potential of S-node be 1. Finally when all of the $s - t$ paths are saturated, that suggests the min-cut is obtained, the extra infeasibility will flow backward to the source node.

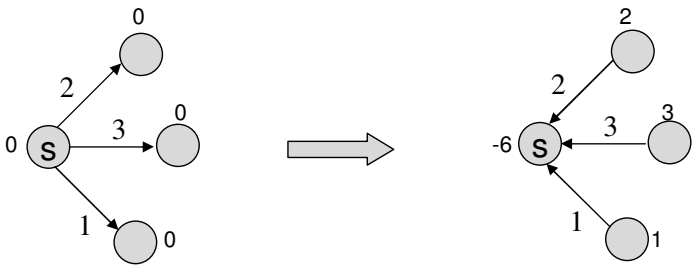

Figure 3.4: Pre-flow push

3.4 Summary

Since the LSPD algorithm has been proved that it is the steepest descent/ascent primal-dual algorithm (Barnes et al., 2002), and network flow problems exhibit the

least-squares property, Gopalakrishnan (2004) proposed the LSNF algorithm that based on the concept of LSPD for solving the MCF problems, the computational results show that the performance of the LSNF algorithm is better than the traditional method, and even much better when we are facing the highly degenerate problems. But the LSNF algorithm can only be applied on the uncapacitated MCF problems. Therefore, we propose the methods to make the LSNF algorithm more comprehensive by solving a BLS problem to acquire the improving direction for a capacitated MCF problem in Section 3.1. We also propose the methods to obtain a dual feasible solution for MCF problems with negative arc lengths Section 3.2, that includes the arc reversal transformation for the capacitated MCF and the label-correcting like technique for the uncapacitated MCF. Since max-flow problems can be viewed as a special case of the MCF problem, we introduced the application of the LSNF algorithm on solving the max-flow problem in Section 3.3.

CHAPTER IV

THE LEAST-SQUARES DUAL-PRIMAL ALGORITHM

In this Chapter, we will introduce the LSDP algorithm, we call it LSDP to differentiate the method from the LSPD algorithm discussed in previous Chapter. Since we mentioned that the network flow problem has the least-squares property, we will show that the dual problem of a network flow problem also has some interesting property that we can make a comparison to the electrical network. We first give an introduction of the LSDP algorithm for the max-flow problem in Section 4.1 which includes the discussion of electrical network problems in Section 4.1.1 and the comparison of NNLS problems in Section 4.1.2. Section 4.2 shows how to apply the LSDP algorithm on solving the MCF problem.

4.1 LSDP for max-flow problem

As we mentioned that we would rather call the primal-dual method applied on max-flow problems as the dual-primal method to differentiate it from the generic primal-dual algorithm since the method solves the main problem at the dual stage, here we will introduce the combinatorial algorithm LSDP for the max-flow problem. The idea is similar to the LSPD algorithm where we replace the RP problem by an NNLS problem to get the improving direction.

First we modify the original flow balance constraints from $Ax + dv = 0$ to $Ax + dv \leq 0$, because a deficit in the flow balance at any node implies a surplus at some other. Since we can take d as an artificial (t, s) arc, we let arc set $A = A \cup (t, s)$ and $|A| = m$. According to the revision, we can formulate the corresponding NNLS problem as follows:

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} s_{ij}^2 \\ s.t. \quad & pA + rI - wI + sI = c \\ & p, r, w \geq 0 \end{aligned} \tag{4.1}$$

where c denotes e_m, $r \in R^m$ and $w \in R^m$ are dual variables related to the capacity and the nonnegative constraints, they are defined by:

$$\begin{cases} r_{ij} \geq 0 \quad \forall (i,j) \text{ such that } x_{ij} = u_{ij} \\ r_{ij} = 0 \quad \text{otherwise} \end{cases}$$

$$\begin{cases} w_{ij} \geq 0 \quad \forall (i,j) \text{ such that } x_{ij} = 0 \\ w_{ij} = 0 \quad \text{otherwise} \end{cases}$$

Now, if we ignore the variables r and w related to the bound constraints of the max-flow problem, we rewrite the NNLS problem as (4.2). The formulation shows an interesting property that the NNLS problem is equal to an electrical network problem as formulated in (4.2) where s denotes the current flow upon the network.

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} s_{ij}^2 \\ s.t. \quad & pA + sI = c \\ & p \geq 0 \end{aligned} \tag{4.2}$$

4.1.1 The electrical network problem

For a simple electrical network which only contains input voltage/current and resistors, we can easily obtain the current and voltage associated with each arc by the Kirchhoff's law which includes the current law and the voltage law.

1. Current Law: It suggests the flow balance constraint that for every node the inflow current will be equal to the outflow current.

2. Voltage Law: For every cycle, the potential difference is zero.

For an electrical network with n nodes and m arcs, there are $2m$ unknown variables comprised by m current variables and m voltage variables of arcs. We have m equations $V = IR$ associated to each arc, and the Kirchhoff's law provides us the rest of m equations including $n-1$ current equations and $m-n+1$ voltage equations to obtain the unique solution. Before we solve an electrical network problem by Kirchhoff's law, we define the direction for each arc, then the negative sign of the solution denotes the current flow in the backward direction of the arc. In fact, an electrical network problem can be solved as a quadratic minimum-cost network problem. Hu (1966) suggested that we can solve an electrical network by formulating a minimum convex-cost network flow problem as below:

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} R_{ij} x_{ij}^2 \\ s.t. \quad & \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = b_i \quad \forall i \in N \end{aligned} \tag{4.3}$$

The variable R_{ij} denotes the resistor attached to arc (i, j), b_i represents the given amount of supply/demand of current of node i, and variable x_{ij} indicates the current flow on arc (i, j). The flow balance constraints are equal to the $n-1$ Kirchhoff's current

law, and the objective function can be interpreted as minimizing the total work done since the work done by arc (i, j) is $R_{ij}x_{ij}^2$. Problem (4.3) is equal to an electrical network, so that the optimal solution should also satisfy the Kirchhoff's voltage law. The fact shows that, besides obeying the current law, the voltage law implies the current will distribute with the best efficiency to produce the minimum electrical work (i.e. $W = VI = RI^2$).

4.1.2 The NNLS problem with arc-node incident matrix

A similar situation can be observed when we solve an NNLS problem related to a max-flow problem. The constraints of (4.2) suggest the Kirchhoff's voltage law. We demonstrate the concept by an example in Figure 4.1. For simplicity, we temporarily ignore the capacity and nonnegative constraints of the max-flow problem.

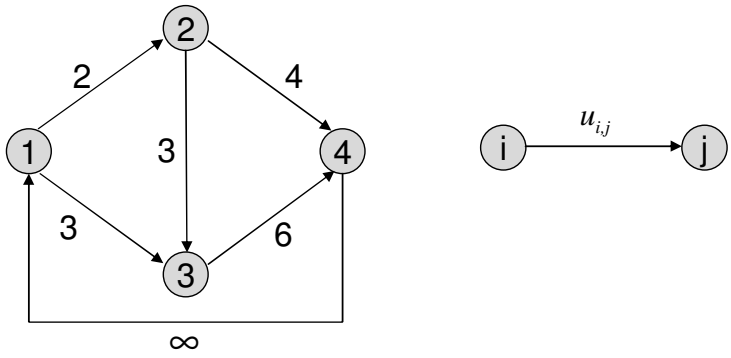

Figure 4.1: A max-flow problem

$$\max \quad x_{41} \tag{4.4}$$

$$s.t. \quad \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{13} \\ x_{23} \\ x_{24} \\ x_{34} \\ x_{41} \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

And the related NNLS problem is as follows:

$$\min \quad \sum_{(i,j)\in A} s_{ij}^2 \tag{4.5}$$

$$s.t. \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} + \begin{bmatrix} s_{12} \\ s_{13} \\ s_{23} \\ s_{24} \\ s_{34} \\ s_{41} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$p_i \geq 0 \ \forall i \in N$$

The constraint matrix of (4.5) is an arc-node incident matrix, so that if we select arcs that can form a cycle, the decision variables p_i will be eliminated, and we have the following conclusion. For each cycle C, equation (4.6) has to be satisfied.

$$\sum_{(i,j)\in C} s_{ij} = b_C = \begin{cases} 1 \text{ if } (t,s) \in C \\ 0 \quad \text{otherwise} \end{cases} \tag{4.6}$$

The constraints are equal to the Kirchhoff's voltage law related to the electrical network in Figure 4.2 where a unit resistor is attached to each arc and (t, s) is

equipped with a unit voltage battery so that the objective of (4.5) minimizes the total electrical work produced and is restricted to the Kirchhoff's voltage law. In addition, since we have mentioned that the improving direction provided by the NNLS problem is feasible, the improving direction s will satisfy the current law which is equal to the flow balance constraints. For the electrical network in Figure 4.2 we can easily observe that the currents/voltages related to arcs $(1,2),(1,3),(2,3),(2,4),(3,4),(4,1)$ are $(\frac{1}{4},\frac{1}{4},0,\frac{1}{4},\frac{1}{4},\frac{2}{4})$ respectively and these are also the optimal slack variables of (4.5). We have the similar conclusion to Hu's model, where the constraints suggest the voltage law and the objective function has the same effect as the current law. If we apply the voltage law, the current obtained by current law will be most efficient to produce

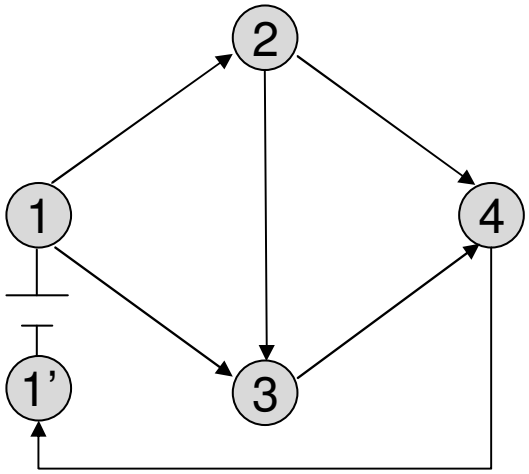

Figure 4.2: An electric network

Before we use the Kirchhoff's law to acquire the optimal slack variables of an NNLS problem, there are some concepts must be discussed in advance. The current upon electrical network may be negative which implies that it flows in the backward direction. We can replace variable r in (4.1)with w by using the arc reversal transformation.

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} s_{ij}^2 \\ s.t. \quad & pA - wI + sI = c \\ & p, w \geq 0 \end{aligned} \tag{4.7}$$

By (4.7), we know that if arc (i, j) is at its lower bound, the corresponding dual variable $w_{ij} \geq 0$, so that while the NNLS is optimized the related optimal slack variable s_{ij} must be nonnegative since the negative value should be taken off by $-w_{ij}$ to minimize the objective value. We have mentioned that (4.2) is equal to the electrical network problem, that we ignore variable w which is related to the capacity and nonnegative constraints. In fact, (4.7) can also be compared to an electrical network except that there is no negative value for arcs with zero flow. In other words, it only allows currents to flow in the forward direction upon certain of arcs. The condition can also be compared to the electrical component, diode, which restricts current to flow in a prespecified direction so that the arc with diode will be disconnected, while the electromotive force (EMF) is opposite to the prespecified direction.

The method that we use to solve an electrical network with diodes is as follows. We let $I = \{(i, j) : s_{ij} \text{ is infeasible}\}$. Since each disconnection will cause the alteration of current and node potential, we can not disconnect all of arc $(i, j) \in I$ simultaneously. On the contrary, we disconnect $(i, j) \in I$ sequentially, and the arc with the most infeasible current will be on the top priority since we think it will have more effects to the entire electrical network. Later, while we detect that a diode which is disconnected in former iteration have inverse EMF, we will connect the arc again.

The procedure of the LSDP algorithm can be summarized as follows. At each iteration, we update the residual network to verify whether a current is feasible or not.

If not, we will adjust the network to obtain the feasible current. A feasible current flows in the backward direction on the saturated arc or forward on a zero-flow arc. Since we make use of the arc reversal technique, we only check these arcs at their lower bounds. Those arcs with infeasible current will be disconnected. As the min-cut is saturated finally, the electrical network will form a broken circuit from s to t by disconnecting arcs with infeasible currents. Therefore when there is no current on the network anymore, the NNLS is optimized with zero objective value, and the optimality condition is attained.

Algorithm 4 NNLS

Input a network G and current s
$D = \emptyset$
While (the current s is not feasible) **Do**
    $I = \{(i,j) : s_{ij}$ is infeasible; $(i,j) \in A\}$
    $s_{ij} = max_{(i,j)\in I}|s_{ij}|$
    $G = G \setminus (i,j)$;
    $D = D \cup (i,j)$;
    solve the electrical network G
    While$(p_i > p_j : (i,j) \in D)$ **Do**
        $G = G \cup (i,j)$
        $D = D \setminus (i,j)$
        solve the electrical network G
    End While
End While

Algorithm 5 The LSDP for the max-flow problem

Input a network G
$x := 0$
Do
    get s by solving the electrical network G
    NNLS(G, s)
    update x through s
    reverse the saturated arc
While (the current is not zero)

In fact, our method to solve the electrical network with diodes can be related to the phase I algorithm proposed by Leichner (1993). Now we make a further discussion

about the electrical network method by the problem in Figure 4.1. We write the corresponding NNLS problem according to the problem with zero flows. Since each arc is at its lower bound, we rewrite the constraints of (4.5) by taking variable w in the consideration.

$$\min \sum_{(i,j)\in A} s_{ij} \tag{4.8}$$

$$\text{s.t.} \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} w_{12} \\ w_{13} \\ w_{23} \\ w_{24} \\ w_{34} \\ w_{41} \end{bmatrix} +$$

$$\begin{bmatrix} s_{12} \\ s_{13} \\ s_{23} \\ s_{24} \\ s_{34} \\ s_{41} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$p_i \; w_{ij} \geq 0 \;\; \forall \; i \in N \; ; (i,j) \in A$$

According to Leichner's (1993) phase I algorithm which solves an NNLS problem, we combine vectors with weights p_i and w_{ij} to minimize the length of the slack vector s. We define a parameter Δ, Δ^2 denotes the improvement after taking a new vector into the basis. The Δ of a nonbasic vector A_i is defined as:

$$\Delta_i = \frac{s^T A_i}{((1-(A_i^T(b-s))^2)/(b-s)^2)^{1/2}} \tag{4.9}$$

At each iteration we select a nonbasic vector A_i such that $s^T A_i > 0$ to let the weights to be nonnegative and has the largest Δ to reduce the infeasibility most. After we get a new vector into the basis, the inner loop of the phase I algorithm solves the least-squares problem, then those vectors with negative weight will be dropped, the inner loop will not terminate till the weight are all nonnegative.

For an NNLS problem in a similar form of (4.8) which is restricted to the arc-node incident constraints, due to the special structure, columns associated to p (node potentials) are always in the basis. Therefore, we can first deal with these columns. After we deal with columns associated to p, we decide which column with weight w to get into the basis. These columns are $-e$ so that $-s^T e_{ij} = -s_{ij}$. As a result, while $s_{ij} < 0$, $-e_{ij}$ is eligible to get into the basis. Furthermore, since b is e_{41}, $(b-s)$ is:

$$b - s = \begin{bmatrix} -s_{12} \\ -s_{13} \\ .. \\ .. \\ 1 - s_{41} \end{bmatrix} \tag{4.10}$$

Therefore, we have the conclusion that $-s^T e_{ij} > -s^T e_{jk}$ and $(-e_{ij}^T(b-s))^2 > (-e_{jk}^T(b-s))^2$ while $s_{ij} < s_{jk} < 0$ for $(i,j) \in A \setminus (t,s)$, since arc (t,s) is never bounded by the lower bound or upper bound except in the beginning, we do not consider to take $-e_{ts}$ as an entering variable. Consequently, if $s_{ij} < s_{jk} < 0$ we have $\Delta_{ij} > \Delta_{jk}$ and that will lead to the precedence of column $-e_{ij}$ to enter the basis. The conclusion is equal to our method that we cut the arc with infeasible current most.

As we mentioned, we compare an NNLS problem similar to (4.8) corresponding

to an electrical network with diodes. And the method we use to solve the electrical network with diodes can be related to Leichner's method which is an effective solution method of NNLS problem. We first solve the electrical network without diodes, and that is equal to solving the (4.5) by ignoring the dual variable w of nonnegative constraints. Then we start to cut arc (i, j) with infeasible current most, that is equal to the entry of $-e_{ij}$ of Leichner's method. Likewise, the reconnection of an arc (i, j) implies $-e_{ij}$ leaves the basis. In fact, each recalculation of current and node potential after the alteration of network structure solves the least-squares problem during the phase I algorithm.

Now we demonstrate the LSDP algorithm by the problem in Figure 4.3. For each iteration, we update the residual network according to the intermediate solution, and the corresponding electrical network is on the right. We obtain the current which is equal to the slack vector of the related NNLS problem to be the improving direction for the max-flow problem. At iteration 1 we have $x = (0, 0, 0, 0, 0, 0)$ and current upon the electrical network is $(s_{12}, s_{13}, s_{23}, s_{24}, s_{34}, s_{41}) = (\frac{1}{4}, \frac{1}{4}, 0, \frac{1}{4}, \frac{1}{4}, \frac{2}{4})$. We push the flow throughout the network until x_{12} reaches the upper bound, then we reverse arc $(1, 2)$. After we reverse arc $(1, 2)$, the current distribution becomes $(\frac{-1}{4}, \frac{1}{4}, 0, \frac{1}{4}, \frac{1}{4}, \frac{2}{4})$. We know that $\frac{-1}{4}$ flow on arc $(1, 2)$ is not allowed, therefore arc $(1, 2)$ is disconnected, the new electrical network will have current distribution $(\frac{3}{8}, \frac{-1}{8}, \frac{1}{8}, \frac{2}{8}, \frac{3}{8})$. Likewise, $\frac{-1}{8}$ flow on arc $(2, 3)$ is infeasible so that we disconnect arc $(2, 3)$ as well, and recalculate the currents. Finally we have current distribution $(\frac{1}{3}, 0, \frac{1}{3}, \frac{1}{3})$ which is a feasible direction to improve the flow on arc $(1, 3)$, $(2, 4)$, $(3, 4)$, $(4, 1)$ till arc $(1, 3)$ is bounded, then we reverse arc $(1, 3)$. Similar principle will be used to obtain the improving direction at the following process, and finally we will find the electrical network becomes a broke circuit, so that there is no current flow upon the electrical network. The max-flow

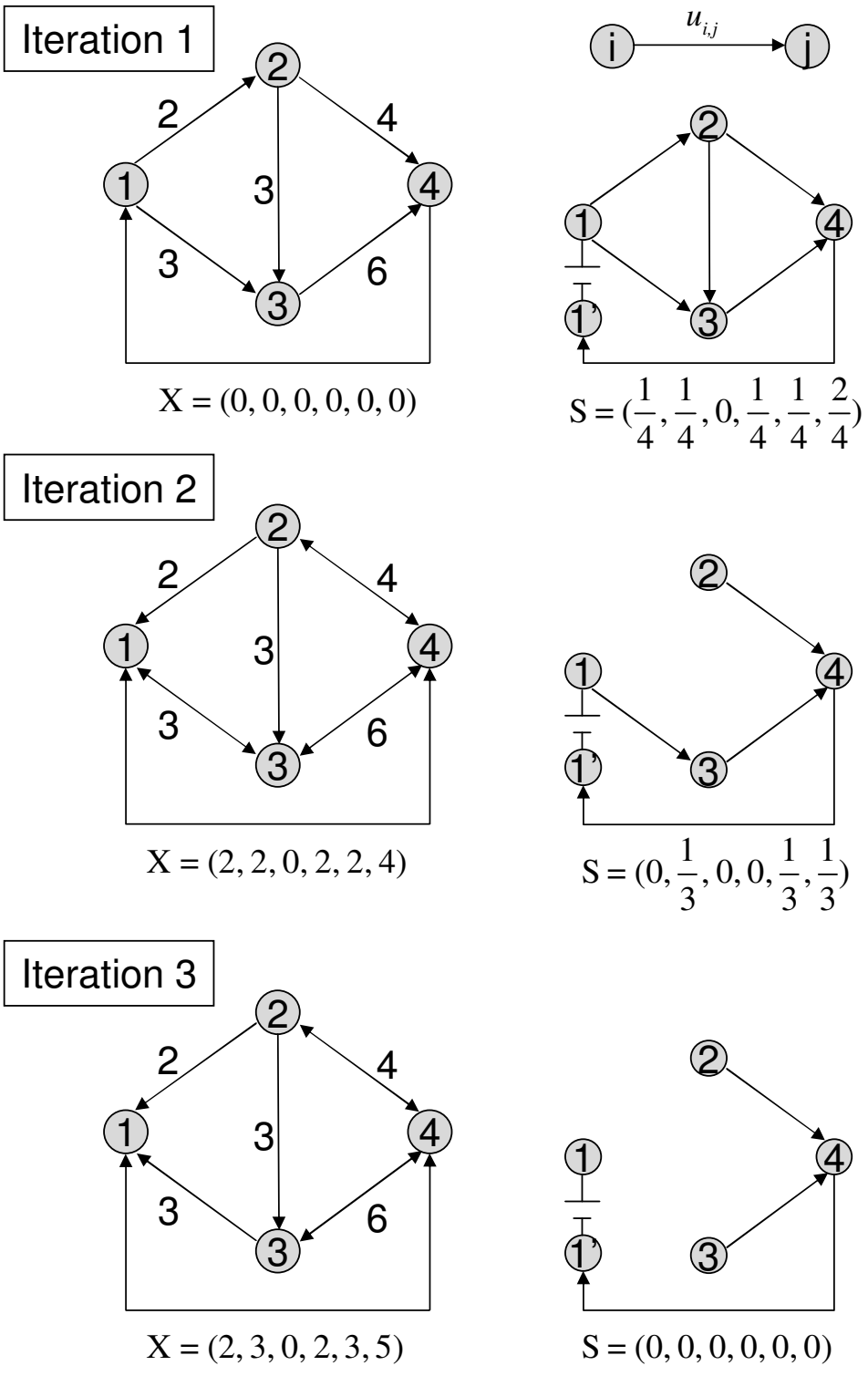

Figure 4.3: An example of LSNF for max-flow problem

problem is optimized with 5 units of flow.

A feature of the LSDP algorithm is that the integer solution property of the network flow problem seems not valid. In fact the optimal solution definitely identify the min-cut, since most of max-flow problems have multi optimal solution, the major target is to find the min-cut. For the LSDP algorithm, after obtaining the optimal solution, we can make use of flow decomposition to cancel some cycle flow to adjust the solution to be integer.

4.2 LSDP for MCF

The LSDP algorithm can also be applied for solving MCF problems. The difference is that b_C in (4.6) is not necessarily zero as it does not contain arc (t, s) which is explained as a unit voltage battery. In fact, the arc length can be viewed as a battery with $-c$ voltage. So that, in the MCF problem, for each cycle, the b_C becomes the summation of arc lengthes in the backward direction minus the summation of those in the forward direction, the voltage law is still valid and so is current law. In Figure 4.4, we attach a battery with $-c$ voltage to each arc, the potential difference of the cycle is 4, so that the 1 unit current flows along the direction of $1 - 2 - 3 - 4 - 1$. This suggests the idea of cycle-canceling. Therefore, if we want to solve an MCF problem with LSDP algorithm, in the beginning, we can obtain an initial feasible solution by adding an artificial (s, t) arc with cost M, and then continue to cancel negative cycles until optimality conditon is attained.

We demonstrate the idea by a capacitated MCF problem in Figure 4.5. For simplicity, we represent the battery with the original arc rather than the additional arc such as arc $(1', 1)$ in Figure 4.3. 3 units of flow will be transported from node 1 to node 4, and the arc lengthes are interpreted as negative voltages. We let the length of the artificial arc (s, t) be 20 which is much larger than the others'. In the beginning,

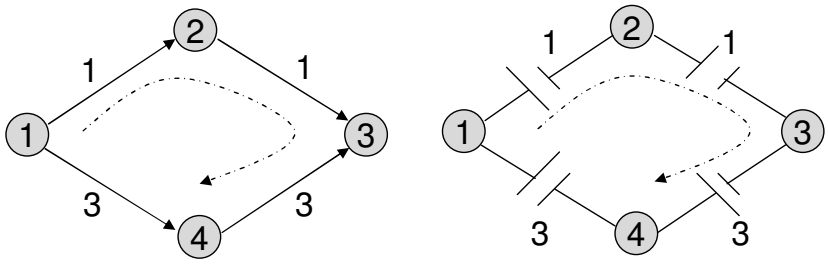

Figure 4.4: A cycle with batteries

we have a feasible solution by pushing flows along the artificial (s,t) arc. We calculate the current distribution of the electrical network is $(4\frac{3}{4}, 3\frac{1}{4}, \frac{-2}{4}, 5\frac{1}{4}, 2\frac{3}{4}, -8)$ associated to arcs $(1,2),(1,3),(2,3),(2,4),(3,4),(1,4)$. We know that $\frac{-2}{4}$ flow on arc $(2,3)$ is not feasible, so we remove arc $(2,3)$ from the electrical network and recalculate the related current. According to the residual network, we have the corresponding electrical network with current $(5,3,0,5,3,-8)$ upon arcs $(1,2),(1,3),(2,4),(3,4),(1,4)$ respectively which are all feasible so that we can improve our MCF problem by pushing flows on arcs $(1,2),(1,3),(2,4),(3,4),(1,4)$ with the proportion of $(5,3,0,5,3,-8)$. Likewise, after we update the residual network, we can calculate the corresponding electrical network and the current is 1 unit of flow along cycle $1-2-4-3$ at iteration 2. At the final iteration, we find that there is no negative cycle on the residual network, so that the corresponding electrical network is a broken circle. Finally, we have the optimal solution $(3,0,1,2,1)$ with objective value 9.

For a some-to-some MCF problem, we can turn it into a one-to-one problem by adding dummy nodes. As shown in Figure 4.6, a brand new dummy supply/demand node with arcs connect to the original supply/demand nodes takes over all of the supply/demand, these arcs have zero arc lengths and their capacities are equal to the amounts of supply/demand of node they connect to.

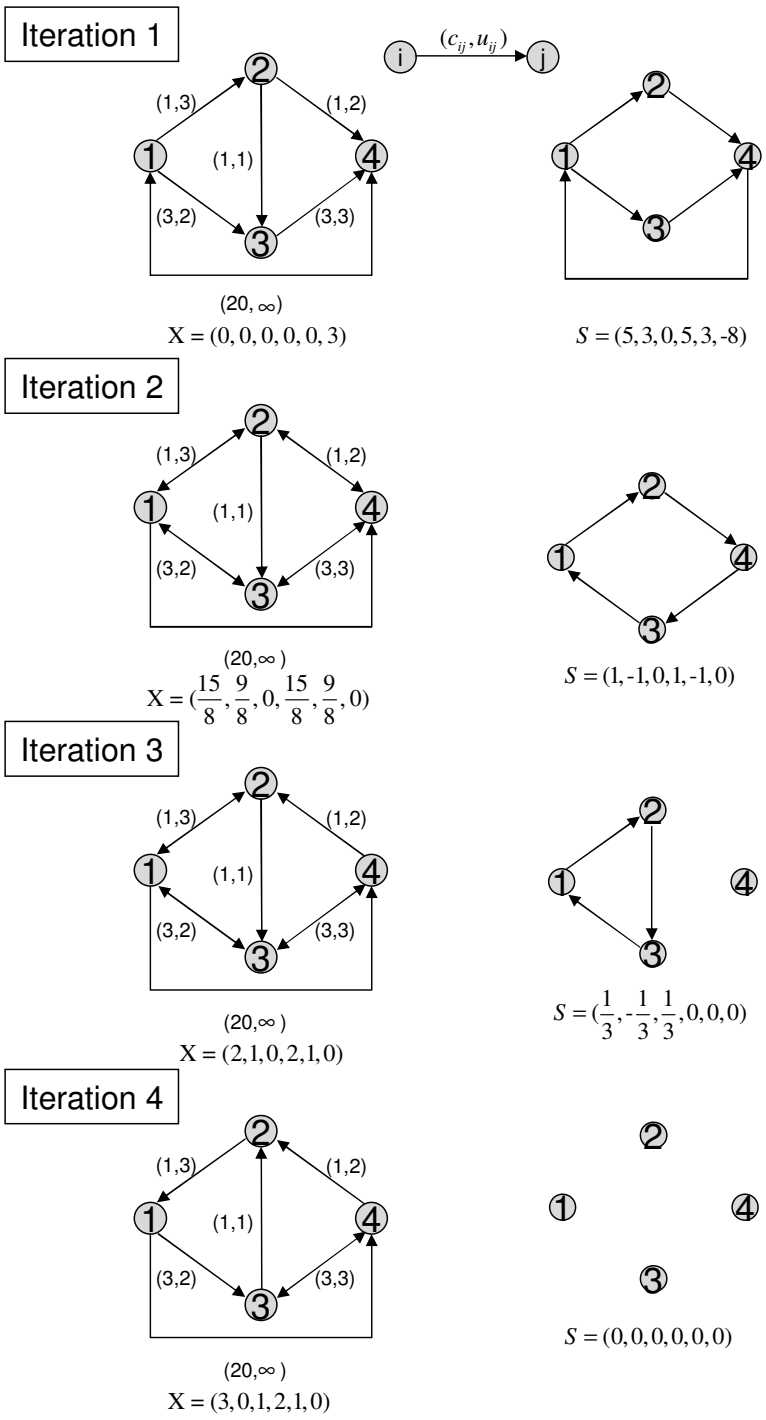

Figure 4.5: LSDP for MCF

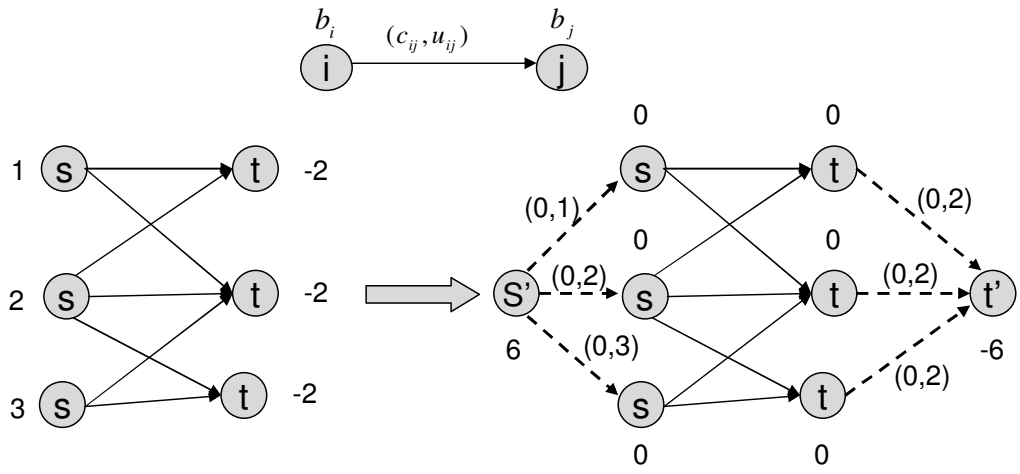

Figure 4.6: Dummy nodes of MCF

4.3 Summary

Due to the special structure of the max-flow problem, the optimal slack variables of NNLS problem with arc-node incident constraint matrix will follow the Kirchhoff's law where each arc is attached to an unit resistance. The procedure of the phase I algorithm can be related to an electric network with diodes which may cause some arcs to be disconnected. Therefore, we can use the Kirchhoff's law to calculate the optimal solution of the NNLS problem to solve the max-flow problem and MCF problem. The application on max-flow problems and the comparison of electrical network and the phase I algorithm have been introduced in Section 4.1. Section 4.2 introduced the application of the LSDP algorithm for MCF problems.

CHAPTER V

CONCLUSION

With the increasing importance of global logistics, the issues of solving the network flow problems are getting more and more critical. Companies have to arrange the distribution effectively and efficiently to improve the competitiveness, the production process can also be optimized by network flow method. The application of network flow problem is widespread.

This thesis studies on how to optimize the network flow problem by a least-squares primal-dual algorithm, the special structure of the network flow problem is helpful for us to solve the least-squares problem. In Chapter 3 we proposed two methods to extend the power of the LSNF algorithm which focuses on solving the MCF problems, including the label-correcting like technique to obtain an initial dual feasible solution for problems with negative arc length and the application of the arc reversal technique to handle those with capacity constraints. The other contribution is that we applied the LSDP algorithm to solve the max-flow problem and observe that the procedures of solving the NNLS problem can be explained by the analogy of calculating the current over an electrical network with diodes, and the nonnegative constraints corresponds to the current over diodes which has to flow in a prespecified direction.

Although the practical efficiency of the LSDP algorithm that we proposed for the max-flow and MCF problems remains to be evaluated, it should have the potential to be

an efficient algorithm since it has good theoretical advantages in avoiding degenerate pivots at each step. Here we suggest the following topics for future research in the related fields:

1. More efficient implementation and computational experiments for the LSDP algorithm:

   The LSDP algorithm we proposed in Chapter 4 is just a prototype. Although we have exploits the Kirchhoff's law to solve the least-squares problem, how to efficiently compute all the infeasibility require more careful design in the data structure and algorithm.

2. Theoretical complexity for the LSDP algorithm

   Although the LSDP algorithm avoids degenerate pivotings, questions such as how many iterations and how much work required per iteration to obtain an optimal solution remain to be open. We think this is an important and challenging research topic worth pursuing.

3. Application of the LSDP algorithm to solve the multi-commodity problems

   The multi-commodity problem is a special network flow problem. How its special structure may affect the process of calculating the NNLS problem is still unknown.

REFERENCES

Ahuja, R., Orlin, J., Sharma, P. and Sokkalingam, P. A network simplex algorithm with $O(n)$ consecutive degenerate pivots. *Operations research letters*, **30**, 141-148, 2002.

Barnes, E., Chen, V., Gopalakrishnan, B. and Johnson, E. L. A least-squares primal-dual algorithm for solving linear programming problems. *Operations research letters*, **30**, 289-294, 2002.

Bazaraa, M. S., Jarvis, J. J. and Sherali, H. D. *Linear programming and network flows.* John Wiley & Sons Inc., 1990.

Cunningham, W. A network simplex method. *Mathematical programming*, **11**, 105-116, 1976.

Curet, N. Implementation of a steepest-edge primal-dual simplex method for network linear programs. *Annals of Operation Research*, **81**, 251-270, 1998.

Dantzig, G. 1948. *Programming in a linear sructure* (Tech. Rep.). United States Air Force.

Dantzig, G. 1951. Application of the simplex method to a transportation problem. In *Activity analysis of production and allocation* (pp. 359–373). New York, N. Y.: John Wiley & Sons Inc.

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

Edmonds, J. and Karp, R. M. Theoretical improvement in algorithmic efficiency for network flow problems. *Journal of the Association for Computation Machine*, **19**(248-264), 1972.

Ford, L. and Fulkerson, D. *Flows in networks.* Princeton, N.J.: Princeton University Press, 1962.

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

Ford, L. R. and Fulkerson, D. R. Maximal flow through a network. *Canadian Journal of Mathematics*, **8**, 399-404, 1956.

Fulkerson, D. R. and Danzig, G. B. Computation of maximum flow in networks. *Naval Reserach Logistics Quarterly*, **2**, 277-283, 1955.

Goldberg, A. V. and Tarjan, R. E. A new approach to the maximum flow problem. *Journal of the Association for Computation Machine*, **35**, 921-940, 1988.

Gopalakrishnan, B., Barnes, E., Johnson, E. and Sokol, J. 2004. *A least-squares network flow algorithm* (Tech. Rep.).

Hu, T. Minimum-cost flow in convex-cost networks. *Naval research logistics quarterly*, **13**, 1-9, 1966.

Karzanov, A. Determining the maximal flow in a network by the method of preflows. *Soviet Math. Doklady*, **15**, 434–437, 1974.

Leichner, S., Dantzig, G. and Davis, J. A strictly improving linear programming phase 1 algorithm. *Annals of Operatins Research*, **47**, 409-430, 1993.

Paparrizos, K., Samaras, N. and Stephanides, G. A new efficient primal dual simplex algorithm. *Computers & Operations Research*, **30**, 1383-1399, 2003.

Wang, I.-L. 2004. *On solving shortest paths with a least-squares primal-dual algorithm* (Tech. Rep.).