STUDIES ON MULTICOMMODITY FLOWS IN DIRECTED NETWORKS

HIROSHI NAGAMOCHI

KYOTO UNIVERSITY
KYOTO, 606 JAPAN
MARCH, 1988

STUDIES ON MULTICOMMODITY FLOWS IN DIRECTED NETWORKS

HIROSHI NAGAMOCHI

KYOTO UNIVERSITY
KYOTO, 606 JAPAN
MARCH, 1988

STUDIES
ON
MULTICOMMODITY FLOWS IN DIRECTED NETWORKS

by
HIROSHI NAGAMOCHI

Submitted in partial fulfillment of the
requirement for the degree of
DOCTOR OF ENGINEERING
(Applied Mathematics and Physics)

KYOTO UNIVERSITY
KYOTO, 606 JAPAN
MARCH, 1988

PREFACE

Network optimization is one of the most important practical branches of mathematical programming, and is encountered in various engineering fields, especially information processing and operations research. Among various network optimization problems, the multicommodity flow problem often arises when more than one commodity shares each arc in a network. This problem can represent many important problems encountered in a wide variety of applications such as traffic assignment in road or communication networks, production scheduling and routing in VLSI design. From theoretical view point, the multicommodity flow problem may be classified into two categories, linear and nonlinear, according to the types of the cost function to be optimized.

The linear model can be formulated as a specially structured linear program, and can be solved in strongly polynomial time, as recently shown by E.Tardos (1986). In practice, the codes based on the simplex method appear to be faster than others and are currently used, but they are not efficient enough to solve large scale problems encountered in practice. It is therefore desirable to develop efficient

network theoretic algorithms.

The nonlinear network models also have been extensively studied in connection with urban traffic systems, communication systems, and many other practical applications. Some algorithms for the general nonlinear programming problems have been specialized to solve nonlinear network problems.

The objective of this thesis is to develop efficient algorithms for the multicommodity flow problem in directed networks, linear and nonlinear models. The main contribution of this thesis may be classified into the following two categories.

The first category contains graph theoretic algorithms for testing feasibility of the multicommodity flow problem in certain planar directed networks. Tree polynomial time algorithms for classes CB, CS and CU of such networks are developed. Furthermore, the integrality of flows and the max-flow min-cut property are investigated: Classes CB, CS and CU all have integral flow property, but only classes CB and CS have the max-flow min-cut property.

In the second category, the minimization of a nonlinear cost function of the multicommodity flow problem in a general directed network is studied. Relaxation

methods of various types for obtaining optimal solutions are proposed, and compared on the basis of computational experiment.

Importance of efficient algorithms in these areas will be increasing. The author hopes that the work contained in this thesis is helpful for further study in this growing field.

March, 1988

Hiroshi Nagamochi

ACKNOWLEDGEMENTS

The author is heartily grateful to Professor Toshihide Ibaraki of Kyoto University for his enthusiastic guidance, discussion and persistent encouragement. Without his support, none of this work would have been possible. He would also like to thank him for his careful reading of this manuscript and accurate comments.

The author would like to express his sincere appreciation to Professor Toshiharu Hasegawa of Kyoto University for his supervision of this work and continuous encouragement.

The author is also highly indebted to Associate Professor Masao Fukushima of Kyoto University for his invaluable discussion and constant encouragement.

Thanks are in order to Assistant Professor Shigeru Masuyama, Assistant Professor Masamitsu Ohnishi and all his friends and colleagues in Professor Ibaraki's laboratory for their encouragement.

Finally, but not least, the author would like to express his heartiest gratitude to his family for their heartfelt cooperation and encouragement.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Network optimization is one of the most important practical branches of mathematical programming, and is encountered in various engineering fields, especially information processing and operations research. It may be classified into two categories, linear and nonlinear, according to the types of the cost function to be optimized. The linear model can be formulated as a specially structured linear program. The special structure exhibits a useful property that it always possesses an optimal integer solution. This integer property permits a number of important combinatorial problems to be formulated and solved as network flow problems [PAPA 82].

In dealing with combinatorial optimization, we should note that, even if the problems usually have finitely many feasible solutions, the straightforward enumeration of all feasible solutions often requires a prohibitively large amount of computation time. Even problems of moderate sizes often become intractable in the sense of practical computer computation. This necessitates the development of efficient

combinatorial optimization algorithms.

It is difficult, however, to develop an algorithm that always works efficiently for all types of combinatorial optimization problems. Some general solution techniques are known, i.e., integer programming [NEMH 72], dynamic programming [BELL 57, DREY 77] and branch-and-bound methods [LAWL 66, IBAR 76a, IBAR 76b, IBAR 77, IBAR 78]. However, these general techniques are not always effective, i.e., there are certain classes of combinatorial optimization problems that become computationally intractable as the problem size increases. Usually, general purpose techniques are less efficient than special purpose algorithms that are designed for only one class of combinatorial optimization problems.

From this viewpoint, therefore, it is advisable to develop a special purpose algorithm suited for the given problem by exploiting its specific structure. Much effort to develop such algorithms has been made in these two or three decades. Efficient algorithms are now available for some classes of combinatorial optimization problems. It is often the case that such tractable classes of problems are formulated as network flow problems in linear model

[PAPA 82]. Therefore it is important to develop special purpose graph theoretic algorithms, which can be more efficient than the general purpose simplex method for linear programming.

The nonlinear network models also have been extensively studied in connection with urban traffic systems, communication systems, and many other practical applications. Some algorithms for the general nonlinear programming problems are specialized to solve nonlinear network problems. For example, based on the convexity of the objective function, theory of monotropic programming [ROCK 81,84] has been developed.

1.2 COMPUTATIONAL COMPLEXITY

In the theory of combinatorial optimization, performance of an algorithm is evaluated by the amount of computation required to solve given problem instances. To analyze this, we shall formalize the size L of each problem instance, and describe the amount of computation as a function of L. The size of a problem instance is usually measured by the length of the input data which is required for its specification. For example, consider a graph $G=(V,E)$. To input G, the set of nodes and the set of edges can be encoded in length $O(|V|+|E|)$, where $|V|$ is the number of nodes and $|E|$ is the number of edges. Here $O(f(x))$, reading order $f(x)$, denotes that it is bounded from above by $cf(x)$ where c is a constant. This notation will be often used in the subsequent discussion. If length $d(v_i,v_j)$ is attached to each edge (v_i,v_j), the length of the input data becomes $O(|E|+|V|) + O(|E|\log d) = O(|V|+|E|\log d)$, where $d=\max\{|d(v_i,v_j)||(v_i,v_j)\in E\}$. The term logd comes from the fact that $\log_2 d$ bits are required to represent an integer d in binary expression [AHO 74, AHO 86].

The time complexity T(L) is a practically important and widely accepted measure of the amount of computation, which is the number of steps required in the computation. The

number of additions, multiplications and comparisons are typical unit operations to count the number of steps. Since it is very difficult to estimate the exact number of steps, its order is mostly discussed.

It should be noted that, in order to determine function T(L) for a given problem size L, we have to consider an infinite number of problem instances with size L. As global measures for these problem instances, the following two have been proposed: average time complexity and worst-case time complexity. The worst-case time complexity guarantees that any problem instance can be solved within that bound. However, it sometimes provides a bound which is too large for most problem instances. The average time complexity is therefore practically more important. However, it is usually very difficult to derive the time complexity averaged over all the possible problem instances. It is also not easy to know the probability distribution of problem instances in the real world. For this reason, this thesis concentrates on the worst-case time complexity.

By an efficient algorithm we mean one whose worst-case time complexity is bounded by a polynomial function of the

input size. The reason for this is that polynomial time complexity increases more slowly with the sizes of problem instances than nonpolynomial, e.g., exponential, time complexity. Recently, the concept of strong polynomiality has been introduced in connection with an open problem related to the minimum cost flow problem. An algorithm is strongly polynomial [TARD 86], if it consists of the (elementary) arithmetic operations, e.g., additions, comparisons, multiplications and divisions, the number of which is polynomially bounded in the dimension of the input. Here the dimension of input is defined to be the number of data items in the input, e.g., the number of arcs and that of nodes in the underlying graph, and the dimension of the matrix in the underlying linear program.

1.3 HISTORICAL BACKGROUND

Efficient algorithms for solving the maximum flow problem, the minimum cost flow problem and the multicommodity flow problem has been extensively studied, from both theoretical and practical viewpoints.

We start with the description of the maximum flow problem of a single commodity. It is formulated as follows, where the network considered is a directed graph (although it can also be defined for an undirected graph [BERG 73]).

N = (G,c): A network.

G = (V,A): A finite directed graph, where V is a set of nodes, and A is a set of arcs.

a(x,y): A directed arc from node x to node y

OUT(x): The set of arcs whose initial node is x.

IN(x): The set of arcs whose terminal node is x.

c: $A \to Z^+$ is a capacity function, where Z^+ is the set of nonnegative integers.

Given a pair of source node s and sink node t in V, it is asked to find a flow f maximizing the flow value g. A flow is a function f: $A \to R$ that satisfies the following conditions (1.1) and (1.2), where f(a) denotes the flow value in arc a.

Flow conservation: For all $x \in V$

$$\sum_{a \in OUT(x)} f(a) - \sum_{b \in IN(x)} f(b) = \begin{cases} g, & \text{if } x = s \\ 0, & \text{if } x \neq s,\ x \neq t \\ -g, & \text{if } x = t. \end{cases} \quad (1.1)$$

Capacity constraint: For all $a \in A$

$$0 \leq f(a) \leq c(a). \quad (1.2)$$

Let $\tilde{g}$ denote the $|V|$ dimensional vector such that the entry corresponding to node s (t) is g (-g), and 0 otherwise. Since constraints (1.1) and (1.2) are linear, we can formulate the maximum flow problem as the following linear program.

maximize: g

subject to $Ax = \tilde{g}$

$0 \leq x \leq c$

where $A=(a_{iu})$, $i \in V$, $u \in A$, is called the node-arc incidence matrix of the graph with size $|V|*|A|$ (a_{iu}=+1 if i is the initial node of arc u, a_{iu}=-1 if i is the terminal node of arc u).

Ford and Fulkerson in their seminal book [FORD 62] developed an algorithm for the maximum flow problem and presented the max-flow min-cut theorem. The first

polynomial algorithm for the maximum flow problem was given by Dinic [DINI 70]. After a number of improvements of the time complexity in the past decade, $O(|V|^3)$ and $O(|A||V|\log|V|)$ are currently known as the best bounds [KARZ 74, SLEA 80, TARJ 83], where $|V|$ is the number of nodes and $|A|$ is the number of arcs in a network.

The following problem is called the minimum cost circulation problem: Find a flow f that minimizes cost $\sum_{a \in A} d(a)f(a)$, where d(a) denotes the cost given to the unit amount of flow in arc a. As in the above case, a flow f satisfies the following conditions (1.3) and (1.4).

Flow conservation:

$$\sum_{a \in OUT(x)} f(a) - \sum_{b \in IN(x)} f(b) = 0, \quad \text{for all } x \in V. \quad (1.3)$$

Capacity constraint:

$$\underline{c}(a) \leq f(a) \leq \bar{c}(a), \quad \text{for all } a \in A, \quad (1.4)$$

where $\underline{c}(a)$ and $\bar{c}(a)$ are lower and upper bounds on the flow in arc a. Obviously, the minimum cost flow problem can also be written as an LP problem in a similar manner [KENN 80].

Edomonds and Karp developed a scaling technique to solve the minimum cost flow problem in polynomial time [EDMO 72]. However, the number of arithmetic operations

required by their algorithm depends on the size of the bounds. Therefore, it is not strongly polynomial. Finding a strongly polynomial algorithm for the minimum cost flow problem has been an open problem for about ten years, until Tardos [TARD 85] answered it affirmatively. Currently, the dual version of the Tardos algorithm and the simplex like method are also known [FUJI 86, ORLI 86, GALI 86].

Since the coefficient matrix in the formulation of the maximum flow problem or the minimum cost flow problem is totally unimodular [BERG 73], all extreme points of the polyhedron defined by the constraint are integers provided that all capacities are integers [HOFF 56].

Multicommodity flow problems arise when more than one commodity share each arc in a network. This problem can represent many important problems encountered in a wide variety of applications such as traffic assignment in roads or communication networks, production scheduling problems and routing in VLSI design.

P: The set of source-sink pairs (s^k,t^k), $k=1,2,\ldots,K$, where K is the number of commodities. We assume that each commodity k has exactly one source s^k and one sink t^k. Let, $S=\{s^k|k=1,\ldots,K\}$ and $T=\{t^k|k=1,\ldots,K\}$

(possibly $s^{k1}=s^{k2}$ or $t^{k1}=t^{k2}$ for $k1\neq k2$),

g: $\{1,2,\ldots,K\} \rightarrow Z^+$, where Z^+ denotes the set of positive integers. Let g^k denote $g(k)$, the amount of supply (=the amount of demand) of commodity k.

The multicommodity flow problem in a directed network is <u>feasible</u> if there exists a set of $f(a,k), a \in A$, $k \in \{1,\ldots,K\}$, which satisfies the following conditions (1.5) and (1.6). Here $f(a,k)$ denotes the flow value of commodity k in arc a.

<u>Flow conservation</u>: For all $x \in V$ and all k

$$\sum_{a \in OUT(x)} f(a,k) - \sum_{b \in IN(x)} f(b,k) = \begin{cases} g^k, & \text{if } x = s^k \\ 0, & \text{if } x \neq s^k,\ x \neq t^k \\ -g^k, & \text{if } x = t^k. \end{cases} \quad (1.5)$$

<u>Capacity constraint</u>: For all $a \in A$

$$\sum_{k=1}^{K} f(a,k) \leq c(a), \quad (1.6)$$

$$f(a,k) \geq 0, \quad k=1,2,\ldots,K.$$

Since the maximum flow problem, the minimum cost flow problem and the multicommodity flow problem can generally be formulated as specially structured linear programming

problems, they can be solved by polynomial time algorithms devised for LP [KHAC 79, KARM 84]. Furthermore, [ORLI 84, TARD 86] showed that if all sizes of entries in the coefficient matrix are bounded by a constant in LP formulation, such LP can be solved in strongly polynomial time. Therefore, we see that the multicommodity flow problem can be solved in strongly polynomial time, though the degree of polynpmial is still very high. In practice, the codes based on the simplex method appear to be faster than others and are currently used. However, the LP formulation often involves a large number of variables and conditions, and such codes based on the simplex method are not efficient enough in many cases. It is therefore desirable to develop efficient network theoretic algorithms [LOMO 85].

Efficient graph theoretic algorithms are known only for very limited classes of undirected networks [HU 69, OKAM 81, OKAM 83, TANG 64]. This perhaps comes from the fact that most of the properties useful for developing efficient algorithms for the single commodity flow problem such as the unimodularity and the max-flow min-cut property can not be directly generalized to the multicommodity flow problem, except for some special cases [FORD 62].

In an undirected network with K=2 commodities, the max-flow min-cut theorem still holds [HU 69] and a polynomial time graph theoretic algorithm is known. Okamura and Seymour [OKAM 81, OKAM 83] have shown that if all sources and sinks are placed only on the boundary of the outer face of a given planar undirected graph, the max-flow min-cut theorem holds for general K. It is known that the minimum cut in a planar network can be obtained by computing the shortest path in the dual network [HASS 81]. Based on these properties, [MATS 85, SUZU 85] developed an efficient algorithm to check its feasibility. The max-flow min-cut theorem is extended for the multicommodity flow in general undirected networks [IRI 70, ONAG 71]. Unfortunately, this extension involves infinite number of conditions.

Contrary to the above results, the max-flow min-cut theorem does not hold for the multicommodity flow problem even with K=2 for directed networks [KENN 78]. Accordingly, not many tractable classes have been known for directed networks in the sence of efficient graph theoretic algorithms. An exception is [DIAZ 72], in which a planar directed network is considered under the assumption that all sources are on the left side of the boundary while all sinks

are on the right side, and furthermore the order of commodities of sources and the order of commodities of sinks appear in the same order.

1.4 OUTLINE OF THE THESIS

In this thesis, we concentrate on the multicommodity flow problem in a directed network. In Chapter 2, we describe basic properties of a directed network, and present notations and definitions necessary for the subsequent discussion. In Chapter 3, we first introduce class CB (capacity balanced networks) of directed planar networks for which it is possible to develop a polynomial time graph theoretic algorithm. Its running time is $O(K|V|)$ for a CB network with K commodities and $|V|$ nodes. It can also be shown that the integral flow property holds for CB. In Chapter 4, we generalize class CB to class CS (capacity semi-balanced networks), and show that CS can be reduced to CB by an $O(|V|)$ time algorithm. Therefore, CS also has a polynomial time graph theoretic algorithm and the integral flow property. In Chapter 5, we introduce class CU (capacity semi-balanced unilateral networks) as another class that is reducible to CB, and developed an $O(|V|^3)$ time algorithm. These classes CS and CU contain certain multi-item multi-stage production scheduling problem [IBAR 82] as a special case, indicating their importance in practical applications. In Chapter 6, we show that the max-flow min-cut theorem holds for classes CB and CS. For this purpose,

we develop an $O(K|V|)$ time algorithm for finding the minimum cut not satisfying the cut-condition if the network is infeasible. In Chapter 7, based on the max-flow min-cut property for classes CB and CS, we present an $O(S(|V|)+|V_B|T(|V|))$ time algorithm for testing feasibility of a problem in CB and CU, where $T(|V|)$ is the time required to obtain the shortest path tree in a planar network with $|V|$ nodes and $S(|V|)$ is its preprocessing time. This algorithm is faster than the one developed in Chapters 3 and 4, when the number of commodities K is large. In Chapter 8, we study the nonlinear multicommodity flow problem that minimizes a strictly convex cost function. Each arc in the network can have lower and upper capacities on individual commodities and on the sum of commodities. By making use of its dual, we formulate the problem as a nonlinear unconstrained optimization problem and propose relaxation methods of various types. Some computational results are also included for various problem instances.

CHAPTER 2

PROPERTIES OF DIRECTED PLANAR NETWORKS

2.1 INTRODUCTION

In this chaper, we describe the basic properties of a directed (planar) network and present notations and definitions necessary for the subsequent discussion in which graph theoretic algorithms are constructed. See [BERG 73] as to other basic terminology in graph theory.

2.2 DEFINITIONS FOR A DIRECTED GRAPH

A node is called a <u>divergent node</u> if it has no entering arc, a <u>convergent node</u> if it has no outgoing arc. When we discuss the connectivity of a graph G, we consider the undirected graph resulting by disregarding the orientation of every arc in A. We define for a pair of sets of nodes X and Y with $X \cap Y=\emptyset$,

$$A(X;Y)\triangleq\{a(x,y)\,|\,x\in X \text{ and } y\in Y\},$$

in paticular $A(X)\triangleq A(X;V-X)$.

A set of arcs C A is called a <u>cut</u> if it is given as

$$C=\{a(x,y)\in A\,|\, x\in X \text{ and } y\in V-X \text{ or } x\in V-X \text{ and } y\in X\}$$

$(=A(X)\cup A(V-X))$ for some X such that $X\neq\emptyset$ and $X\neq V$. A cut C is <u>simple</u> if it does not properly contain any other cut.

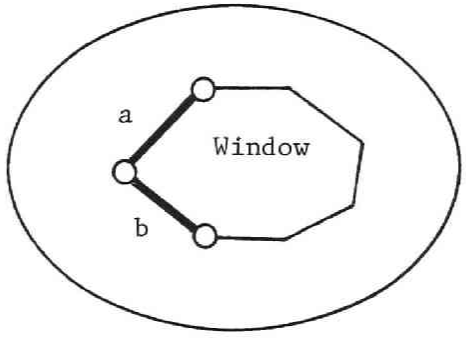


Fig. 2.1 Arcs a and b are immediate neighbours each other.

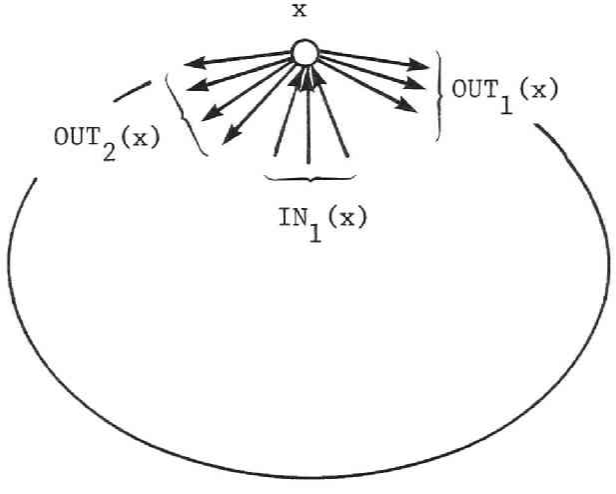


Fig. 2.2 Definition of $IN_i(x)$ and $OUT_i(x)$.

Thus removing all arcs in a simple cut C decomposes a connected graph G into exactly two components. A set of nodes $X \subseteq V$ is called divergent if there is no arc from V-X to X. Similarly X is called convergent if there is no arc from X to V-X. A cut A(X) is called unilateral if X and V-X are divergent and convergent, respectively. A set of nodes X is called connected if the subgraph induced by X is connected.

Let $\Pi(x,y)$ denote the set of all directed paths from x to y. For a $\pi \in \Pi(x,y)$, $V(\pi)$ denotes the set of nodes in π, and $A(\pi)$ denotes the set of arcs in π. We say that a node y is reachable from a node x (i.e., x is reachable to y) if $\Pi(x,y) \neq \emptyset$. If a directed path from x to y and arc a(y,x) exist for some nodes x and y, they consists closed directed path which is called a directed cycle. A directed path (cycle) without repeated nodes is called simple.

Here we consider a planar directed graph G. In subsequent discussion, G is embeded in the plane and we fix it. In G, a cycle which is obtained by ignoring the arc orientation, divides the plane into two areas. A window of G is a simple cycle in G such that one of the resulting two areas contains no arc of G. The boundary B of G is clearly a window. Arcs $a,b \in IN(x) \cup OUT(x)$, are called immediate

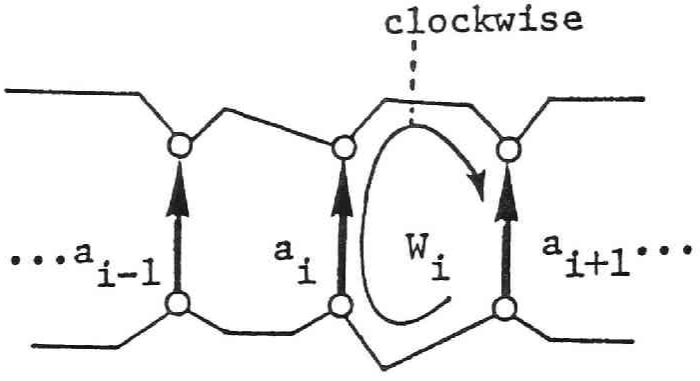


Fig. 2.3 Definition of unilateral chain circuit.

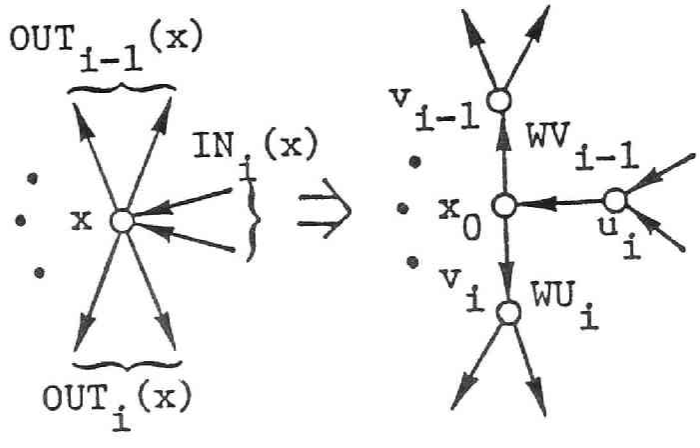


Fig. 2.4 Proof of Lemma 2.5.

neighbours if both a and b are contained in a window which is not the boundary. See Figure 2.1. For each node x, we partition IN(x) and OUT(x) into $IN_i(x)$ and $OUT_j(x)$ respectively as follows. Each of the $IN_i(x)=\{a_0,a_1,\ldots,a_m\}$, $i=1,2,\ldots,i_x$, is a maximal subset of IN(x) with the property that $a_{k-1},a_k \in IN_i(x)$ iff a_{k-1} and a_k are immediate neighbours for $k=1,2,\ldots,m$. Similarly for $OUT_j(x)$, $j=1,\ldots,j_x$. See Figure 2.2.

A sequence $[a_1,a_2,\ldots, a_m]$ of arcs is cutting, if each pair of a_i and a_{i+1} $(i=1,\ldots,m-1)$ is contained in a window W_i, and all W_i's are distinct. Note that one of the windows W_i's may be the boundary B. A cutting sequence $[a_1, a_2,\ldots, a_m]$ is called a cutting circuit, if a_m and a_1 are contained in a window W_m, and all W_i $(i=1,\ldots,m)$ are distinct. A cutting sequence $[a_1, a_2,\ldots, a_m]$ is called unilateral, if the direction of each arc a_i $(i=1,\ldots,m-1)$ is the same as that of window W_i when we go round W_i in the clockwise manner. See Figure.2.3. The above definitions are easily understandable in terms of the directed dual graph $G^*=(V^*,A^*)$ corresponding to G. Each node $v_i^* \in V^*$ corresponds to window W_i in G (boundary B corresponds to v_B^*), and every arc $a_k \in A$ corresponds to exactly one arc $a_k^*=a(v_i^*,v_j^*)$ such that the two distinct windows W_i and W_j

corresponding to v_i^* and v_j^* have common arc a_k in G and the direction of a_k is the same as that of window W_i when we go round W_i in the clockwise manner (counterclockwisely if W_i is the boundary). Then cutting sequences and cutting circuits in G correspond simple paths and simple ciruits (neglecting the orientations) in G^* respectively. It is also clear from definition that if a cutting sequence and a cutting circuit are unilateral, they correspond to a simple directed path and a simple directed circuit in G^* respectively.

2.3 ASSUMPTION A AND SOME LEMMAS

The following Assumption A is important, because all classes CB, CS and CU discussed in Chapters 5-8 satisfy this assumption. we show some properties of a graph G satisfying Assumption A.

Assumption A: (1) G=(V,A) is planar, acyclic (i.e., has no directed circuit). G has no articulation points (i.e., G is 2-connected), where a node is called an articulation point if the number of components in G increases at least by one after deleting the node. Furthermore, we fix a drawing of G in the plane, and define

B: the boundary of the outer face of G,

V_B: the set of nodes in B ,

A_B: the set of arcs whose both end-nodes are in V_B.

(2) Any divergent or convergent node belongs to V_B.

(3) $T \subseteq V_B$. (Recall that sink nodes are not necessarily convergent. The subsequent discussion can be easily modified for the case in which condition (3) is changed to $S \subseteq V_B$. We assume this (3) for simplicity.) □

Lemma 2.1 If $\Pi(x,y) = \emptyset$ for nodes x and y in G, then there are two connected node sets X^-, $X^+ \subseteq V$ such that $x \in X^-$, $y \in X^+$, $X^- \cup X^+ = V$, $X^- \cap X^+ = \emptyset$, X^- is convergent and

X^+ is divergent.

Proof. Let X_T^- be the set of all nodes (including x) reachable from x (i.e., X_T^- is the node set of maximal directed tree T with root x). This X_T^- does not include y, and the direction of every arc between X_T^- and $V-X_T^-$ is from $V-X_T^-$ to X_T^-. Remove the subgraph spanned by X_T^- together with these arcs from $V-X_T^-$ to X_T^-, and we obtain connected components $X_k, k=1,2,\ldots,p$ each of which is divergent. Clearly each X_k is connected to X_T^- (since G is connected) and there are no arcs between X_k and $X_{k'}$ with $k \neq k'$. Name the component X_k containing y by X^+, and let $X^- = V-X^+$. This X^- is therefore connected and convergent. Consequently X^+ and X^- satisfy the lemma's statement. □

Lemma 2.2 In a planar graph G, a set of arcs $\{a_1,a_2,\ldots,a_m\} \subseteq A$ is a simple cut if and only if there is a cutting circuit $[a_{i1},a_{i2},\ldots,a_{im}]$ such that $\{a_{i1},a_{i2},\ldots,a_{im}\}=\{a_1,a_2,\ldots,a_m\}$. Further, a cutting circuit $[a_{i1},a_{i2},\ldots,a_{im}]$ is unilateral if and only if the corresponding simple cut $\{a_1,a_2,\ldots,a_m\}$ is unilateral.

Proof. Let G^* be the directed gragh dual to G. Obviously a simple cut $\{a_1,a_2,\ldots,a_m\}$ in G corresponds a simple cycle (disregarding arc orientation) in G^*. The

lemma follows from the above discussion, because a simple cycle in G^* corresponds to a cutting circuit in G. Unilaterality is clear from definition. □

Lemma 2.3 Assume that G satisfies Assumption A (1) and (2). If a set of nodes X in G is divergent or convergent, then $X \cap V_B \neq \emptyset$.

Proof. Assume that X is divergent. X contains at least one divergent node x_0, otherwise a directed cycle exists in X. Therefore, by (2) of Assumption A, $X \cap V_B \neq \emptyset$ follows. The case of a convergent X is similar. □

Lemma 2.4 Assume that G satisfies Assumption A (1) and (2). Every unilateral cutting circuit C contains exactly two arcs in A_B.

Proof. By lemma 2.2, C divides V into a connected divergent X set and a connected convergent set V-X. By Lemma 2.3, $X \cap V_B \neq \emptyset$ and $(V-X) \cap V_B \neq \emptyset$. Therefore, $C=[a_1, a_2, \ldots, a_m]$ has a_i and a_{i+1}, where window W_i containing a_i and a_{i+1} is the window of the boundary. Since all window W_i, $i=1,2,\ldots,m$ are distinct in cutting circuit C, any other arcs a_j, $j \neq i, i+1$ are not in A_B. □

Lemma 2.5 Corresponding to arc sets $IN_i(x)$ $(i=1,\ldots,i_x)$, and $OUT_j(x)$ $(j=1,\ldots,j_x)$ of a node x in G, there is a partition of V into X_i^+ $(i=1,\ldots,i_x)$, X_j^- $(j=1,\ldots,j_x)$ and $X_0 = \{x\}$ such that all X_i^+ are connected divergent sets, and all X_j^- are connected convergent sets. The set of the terminal nodes of arcs in $OUT_j(x)$ (the set of the initial nodes of arcs in $IN_i(x)$) is contained in X_j^- (X_i^+).

Proof. From the definition of i_x and j_x, we have $i_x=j_x$ if (a) $x \notin V_B$ or (b) $x \in V_B$, $a(z_1,x)$, $a(x,z_2) \in A_B$ for some $z_1,z_2 \in V_B$. We have $i_x=j_x+1$ (or $i_x=j_x-1$) if $x \in V_B$ and $a(z_1,x)$, $a(z_2,x) \in A_B$ (or $a(x,z_1)$, $a(x,z_2) \in A_B$) for some $z_1 \neq z_2$. We shall consider only the case of $x \notin V_B$, since the other case can be treated in a similar manner. Replace node x by the set of node x_0, u_i $(i=1,\ldots,i_x)$ and v_i $(i=1,\ldots,i_x)$, and make u_i the terminal node of the arcs in $IN_i(x)$, and v_i the initial node of the arcs in $OUT_i(x)$. We then add arcs $a(u_i,x_0)$ $(i=1,\ldots,i_x)$ and $a(x_0,v_i)$ $(i=1,\ldots,i_x)$. Let $\hat{G}=(\hat{V},\hat{A})$ be the obtained graph (see Figure 2.4), which still satisfies Assumption A. Now there exists a cutting circuit $[a(u_1,x_0),a(x_0,v_1),\ldots,$ $a(u_i,x_0),a(x_0,v_i),\ldots,$ $a(u_{ix},x_0)$, $a(x_0,v_{ix})]$, where we denote the window containing $a(u_i,x_0)$ and $a(x_0,v_i)$ by WU_i

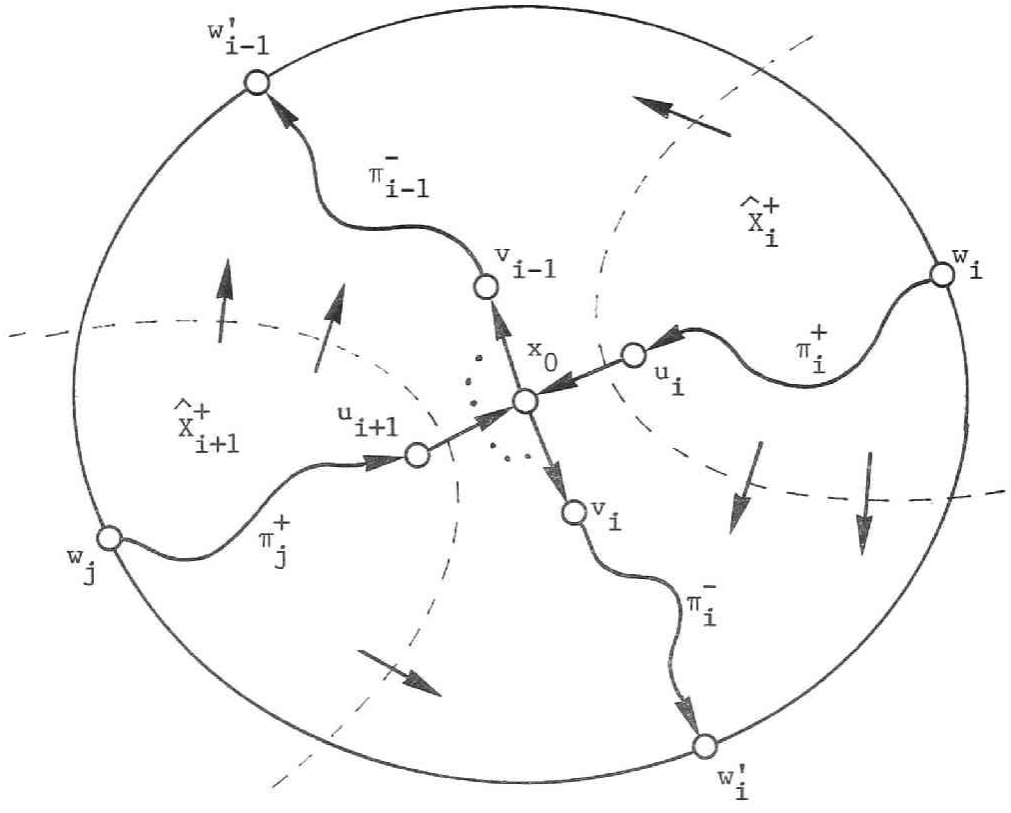


Fig. 2.5 Proof of Lemma 2.5.

and the window containing $a(x_0,v_i)$ and $a(u_{i+1},x_0)$ by WV_i (with the convention $i_x+1=1$). By $\Pi(x_0,u_i)=\emptyset$ (by the acyclicity of $\hat{G}$) and Lemma 2.1, there exists for each i a divergent set $\hat{X}_i^+$ containing all y such that $a(y,u_i)\in\hat{A}$. Let $A(\hat{X}_i^+)$ be the arcs between $\hat{X}_i^+$ and $V-\hat{X}_i^+$. We show that these $A(X_i^+)$, $i=1,2,\ldots,i_x$ are disjoint as shown in Figure 2.5 (as the result, it follows that $\hat{X}_i^+$, $i=1,2,\ldots,i_x$ are disjoint). By (2) of Assumption 2.1, there exist for each i a node $w_i\in\hat{V}_B$ and a path $\pi_i^+\in\Pi(w_i,x_0)$ with $a(u_i,x_0)\in A(\pi_i^+)$ (similarly a node $w'_i\in\hat{V}_B$ and a path $\pi_i^-\in\Pi(x_0,w'_i)$ with $a(x_0,v_i)\in A(\pi_i^-)$). Clearly, by the acyclicity of $\hat{G}$, each π_i^- $(i=1,2,\ldots,i_x)$ are node disjoint to any of π_i^+ $(i=1,2,\ldots,i_x)$ except node x_0. Since each $\hat{X}_i^+$ is contained in the area bordered by π_{i-1}^- and π_i^- (with the convention that $i-1=i_x$ for $i=1$), and $A(\hat{X}_i^+)$ does not contain any arcs of paths π_{i-1}^- or π_i^- (by the divergency of $\hat{X}_i^+$), all $A(\hat{X}_i^+)$, $i=1,2,\ldots,i_x$ have no common arc and all $\hat{X}_i^+$ are node disjoint.

Note that each $\hat{V}-\hat{X}_i^+$ is connected (from Lemma 2.1 and construction of $\hat{X}_i^+$), then we have unilateral cutting circuits $[a_1,\ldots,a_m]$ and $[b_1,\ldots,b_n]$ such that $A(\hat{X}_i^+)=\{a_1,\ldots,a_m\}$, $A(\hat{X}_{i+1}^+)=\{b_1,\ldots,b_n\}$ and $a_1,a_m,b_1,b_n\in\hat{A}_B$ $(=A_B)$ (by Lemma 2.4). Recall that $a_p=a(u_i,x_0)\in A(\hat{X}_i^+)$

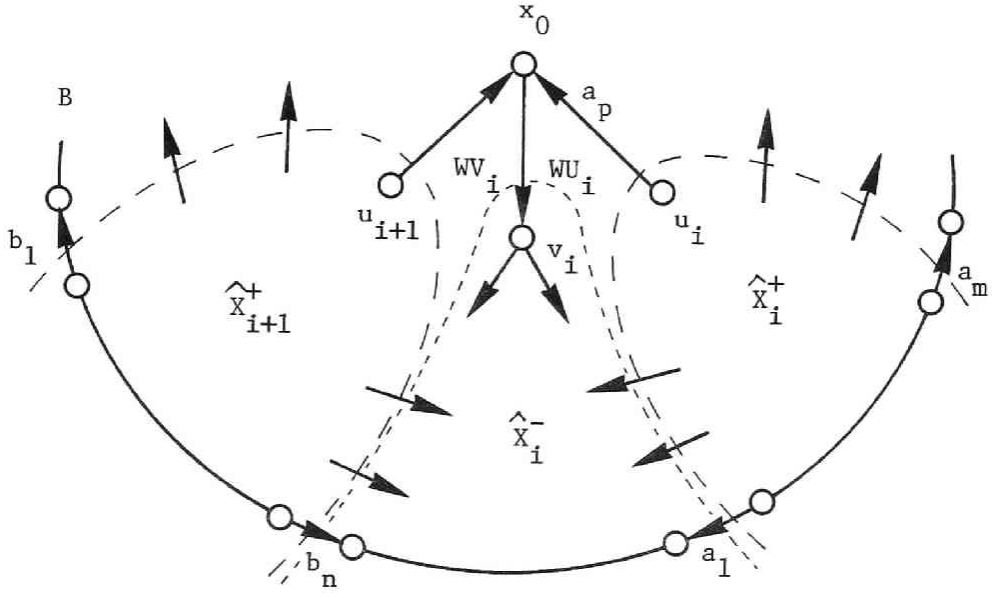


Fig. 2.6 Proof of Lemma 2.5.

and $b_q=a(u_{i+1},x_0) \in A(\hat{X}_{i+1}^+)$ for some p and q. We can assume that window WU_i contains $a(x_0,v_i),a_p$ and a_{p-1}, and WV_i contains $a(x_0,v_i),b_q$ and b_{q+1}, without loss of generality. Here, no window contains $a_i,a_{i+1}(i+1\leq p-1)$ and $b_j,b_{j+1}(j\leq q+1)$. Otherwise,

$$[a_{i+1},a_{i+2},\ldots,a_{p-1},a(x_0,v_i),b_{q+1},b_{q+2},\ldots,b_j]$$

corresponds a directed cycle in graph $\hat{G}^*$ dual to $\hat{G}$, and a simple directed cycle contained in this directed cycle corresponds a unilateral cutting circiut in $\hat{G}$. However, since this unilateral cutting circuit does not contain arc (a_1,b_n) in the boundary, this contradicts Lemma 2.3. Therefore,

$$[a_1,a_2,\ldots,a_{p-1},a(x_0,v_i),b_{q+1},b_{q+2},\ldots,b_n]$$

is a unilateral cutting circuit. Apply Lemma 2.2 to this cutting circuit, and denote the component containing v_i by $\hat{X}_i^-$ as shown in Figure 2.6. In this way, the node set of $\hat{G}$ is partitioned into $\{x_0\}$, connected divergent sets $\hat{X}_i^+$ $(i=1,\ldots,i_x)$ and connected convergent sets $\hat{X}_i^-$ $(i=1,\ldots,i_x)$.

Finally it is easy to see that this partition of $\hat{G}$ gives the stated partition of G, i.e., let $X_i^+=\hat{X}_i^+-\{u_i\}$ and

$X_i^- = \hat{X}_i^- - \{v_i\}$. Each of the resulting X_i^+ (X_i^-) is connected because if some $\hat{X}_i^+$ ($\hat{X}_i^-$) becomes unconnected by deleting u_i (v_i) then G becomes unconnected by deleting x_0, contradicting the 2-connectivity of G. Therefore the resulting family of node sets is the partition of G. □

Lemma 2.6 In a graph satisfying (1) and (2) of Assumption A, let G" be the graph obtained by shrinking X^+ (or $V-X^+$) into node x_0 for any unilateral simple cut $A(X^+)$. Then

(i) G" still satisfies (1) and (2) of Assumption A.

(ii) For any nodes x,y ($\neq x_0$) in G", $\Pi(x,y)\neq\emptyset$ in G implies $\Pi(x,y)\neq\emptyset$ in G".

(iii) Any simple cut in G" is a simple cut in G.

Proof. We consider the case X^+ is shrunken into node x_0 (similarly for $V-X^-$). First we show (i). The planarity of G is obvious. If G" has a directed cycle C, then C does not contain any arc incident to divergent node x_0 and therefore the C exists in G. By the acyclicity of G, this means that G" is acyclic, a contradiction. The 2-connectivity of G" is shown as follows. G is clearly connected. Then if G" is not 2-connected, then G" has an articulation point z. By the 2-connectivity of G, there exist two node-disjoint undirected paths p_1, p_2 between any

two nodes $x,y \notin X^+$. Since there still exist two node-disjont undirected paths between x and y if one of p_1 and p_2 contain no node in X^+, $z=x_0$ must hold. However, by definition, $A(X^+)$ is a uniulateral simple cut, and therefore the resulting graph obtained by removing x_0 from G" is connected. This contradicts that x_0 is an articulation point.

Next for any $x,y \notin X^+$, there is no path $\pi \in \Pi(x,y)$ through X^+ or x_0 in G or G", respectively. Therefore we obtain (ii). (iii) is also obvious from the definition of G". □

2.4 CUT CONDITION AND UNILATERAL CUT

For two sets of nodes X and Y with $X \cap Y = \emptyset$ in $N=(G,P,g,c)$, we define

$$c(X;Y) \triangleq \sum_{a \in A(X;Y)} c(a), \text{ in particular } c(X) \triangleq c(X;V-X),$$

$$K(X;Y) \triangleq \{k \mid s^k \in X, t^k \in Y\}, \text{ in paticular } K(X) \triangleq K(X;V-X),$$

$$g(X;Y) \triangleq \sum_{k \in K(X;Y)} g^k, \text{ in particular } g(X) \triangleq g(X;V-X),$$

$$r(X;Y) \triangleq c(X;Y) - g(X;Y), \text{ in particular } r(X) \triangleq r(X;V-X).$$

Here the cut-condition for the multicommodity flow problem:

$$r(X) \geq 0 \qquad \text{for all } X \subseteq V \tag{2.1}$$

is cleary a necessary condition for a network to be feasible. In general, however, the cut-condition is not sufficient to guarantee the feasibility of the multicommodity flow problem. If the cut-condition is a sufficient condition for a network in some class to be feasible, we call the max-flow min-cut theorem holds for the class (or the class has the max-flow min-cut property).

If a network $N=(G,P,g,c)$ satisfying Assumption A has a pair $(s^{k1},t^{k1}) \in P$ with $\Pi(s^{k1},t^{k1})=\emptyset$, then N is clearly infeasible (by assumption $g^k>0$). In this case, Lemma 2.1

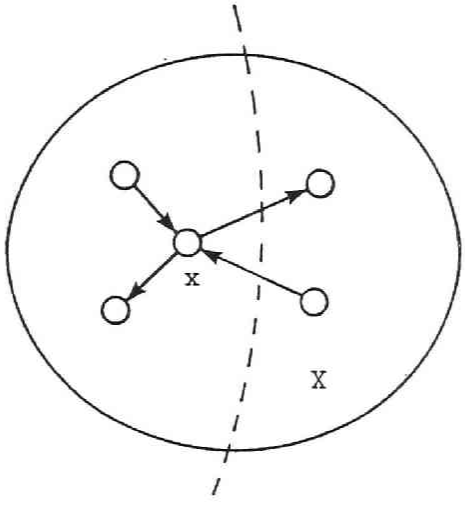


Fig. 2.7 Proof of Lemma 2.7.

implies that N has a connected convergent set X^- and a connected divergent set $V-X^-$ such that $s^{kl} \in X^-$ and $t^{kl} \in V-X^-$. Then

$$r(X^-) = c(X^-)-g(X^-) = -g(X^-) \leq -g^{kl}<0$$

holds, i.e., $A(V-X^-)$ is a unilateral simple cut not satisfying the cut-condition. Therefore, the max-flow min-cut theorem holds.

Based on this, we concentrate in the subsequent discussion on a network that satisfies the following Assumption B.

Assumption B: $\Pi(s^k,t^k)\neq\emptyset$ for every $(s^k,t^k) \in P$. □

Now we introduce the following capacity balance function $\Delta c(x)$ and Assumption C.

$$\Delta c(x) \triangleq \sum_{a \in OUT(x)} c(a) + \sum_{t^k=x} g^k - \sum_{b \in IN(x)} c(b) - \sum_{s^k=x} g^k.$$

Assumption C: $\Delta c(x)=0$ for every node $x \in V$. □

Under this assumption, the following lemma tells that any unilateral cut $A(X^+)$ satisfies the cut-condition (2.1).

Lemma 2.7 In a network N,

$$\sum_{y \in X} \Delta c(y) = r(X) - r(V-X) \quad \text{for } X \subseteq V.$$

Proof. For $X=\{x\}$,

$$r(X) = \sum_{a \in OUT(x)} c(a) - \sum_{s^k=x} g^k \quad \text{and} \quad r(V-X) = \sum_{a \in IN(x)} c(a) - \sum_{t^k=x} g^k$$

hold, i.e., $r(X)-r(V-X) = \Delta c(x)$ satisfying the lemma's statement. Now, we show that $X'=X \cup \{x\}$ satisfies $r(X')-r(V-X') = \sum_{y \in X'} \Delta c(y)$ for any $x \in V-X$, if $r(X)-r(V-X) = \sum_{y \in X} \Delta c(y)$ for some X. See Figure 2.7.

$$\begin{aligned} c(X') &= c(X;V-X')+c(\{x\};V-X') \\ &= c(X)-c(X;\{x\})+c(\{x\};V-X'), \\ c(V-X') &= c(V-X';X)+c(V-X';\{x\}) \\ &= c(V-X)-c(\{x\};X)+c(V-X'+\{x\}). \end{aligned}$$

Hence,

$$c(X')-c(V-X') = c(X)-c(V-X)+ \sum_{a\ OUT(X)} c(a) - \sum_{a\ IN(x)} c(a).$$

Similarly we have

$$g(X')-g(V-X') = g(X)-g(V-X)+ \sum_{t^k=x} g^k - \sum_{s^k=x} g^k.$$

Then

$$\begin{aligned} r(X')-r(V-X') &= c(X')-c(V-X')-g(X')+g(V-X') \\ &= r(X)-r(V-X)+ \Delta c(x) \end{aligned}$$

holds. Therefore by induction, we obtain $r(X)-r(V-X)=$

$\sum_{y \in X} \Delta c(y)$ for any $X \subseteq V$. □

By this lemma, if N satisfies Assumption C, then $r(X)-r(V-X)=0$ for all $X \subseteq V$.

Lemma 2.8 If a network satisfies Assumption B and C, then any divergent set X^+ (unilateral cut $A(X^+)$) satisfies $r(X^+)=r(V-X^+)=0$ (i.e., satisfying the cut condition (2.1)).

Proof. Take a unilateral cut $A(X^+)$ with $r(X^+)<0$. By Lemma 2.7 and Assumption C, $r(X^+)=r(V-X^+)$, and then $r(X^+)<0$ implies $r(V-X^+)<0$. From divergency of X^+, $c(V-X^+)=0$. Hence

$$r(V-X^+)=c(V-X^+)-g(V-X^+)=-g(V-X^+)<0$$

holds, and this means $K(V-X^+)\neq\emptyset$. For $k \in K(V-X^+)$, however $\Pi(s^k,t^k)=\emptyset$ from convergency of $V-X^+$. This contradicts Assumption B, and we obtain $r(X^+)\geq 0$. Here assume $r(X^+)>0$, then we have $g(V-X^+)<0$ in the same manner. This contradicts the assumption $g^k>0$. Therefore we obtain $r(X^+)=r(V-X^+)=0$.

□

CHAPTER 3

ALGORITHM ASSIGN FOR TESTING FEASIBILITY OF A CB NETWORK

3.1 INTRODUCTION

In this chapter, we introduce class CB (capacity balanced networks) of directed planar networks for which it is possible to develop a polynomial time graph theoretic algorithm. Its running time is $O(K|V|)$ for a CB network with K commodities and $|V|$ nodes. It is also be shown that integral flow property holds for CB, i.e., an integral feasible flow exists if the network is feasible and capacities of arcs are all integers [NAGA 87a].

3.2 CAPACITY BALANCED NETWORKS

A network $N=(G,P,g,c)$ is called capacity balanced if it satisfies Assumption A and C, and the class of capacity balanced networks is denoted by CB.

In order to present algorithm ASSIGN that tests the feasiblity of the multicommodity flow problem of (1.5) and (1.6) for a capacity balanced network $N=(G,P,g,c)$, we prepare the following notations.

By the acyclicity of G, the level of each node x can be defined as follows.

$$\text{level}(x) \triangleq \begin{cases} 0, & \text{if } x \text{ is a divergent node} \\ \max_{v} L(\pi_{vx}), & \text{where } v \text{ runs over all divergent nodes such that } \Pi(v,x)\neq\emptyset, \\ & \text{and } L(\pi_{vx}) \text{ is the number of arcs in a} \\ & \text{simple longest path } \pi_{vx} \in \Pi(x,v). \end{cases}$$

For each $OUT_j(x)$, there is a unilateral cutting sequence $[a_1,a_2,\ldots,a_m]$ such that $OUT_j(x)=\{a_i \mid i=1,2,\ldots,m\}$. We define that a_i is to the left of a_j for $1\leq i<j\leq m$. Given the connected convergent set X_j^- obtained for $OUT_j(x)$ by Lemma 2.5, we define the left-right relation between sinks in $X_j^- \cap V_B$ as follows: Sink t^{k1} is to the left of sink t^{k2}

if we visit t^{k1} before t^{k2} when we go round the part of boundary contained in $X_j^{-} \cap V_B$ in the clockwise manner. In this case, we also say that commodity k1 is to the left of commodity k2. For k1 and k2 with $t^{k1} = t^{k2}$, we define that commodity k1 is to the left of commodity k2, if index k1 is smaller than k2.

3.3 ALGORITHM ASSIGN

We first describe the outline of algorithm ASSIGN. Clearly, a network not satisfying Assumption B is infeasible. ASSIGN chooses nodes x in the nondecreasing order of their levels, and, for each of the chosen nodes, it determines the flows in the arcs in OUT(x). When a node x is chosen, the flows on IN(x) are already known, since the flows on OUT(y) of all nodes y with level(y)<level(x) have already been determined. For each of $j=1,\ldots,j_x$, Lemma 2.5 asserts that there exists a connected convergent set X_j^- corresponding to $OUT_j(x)$. Let

$$K_j(x) \triangleq \{k \mid t^k \text{ is reachable from } x \text{ via some arcs in } OUT_j(x)\}.$$

Clearly $\{t^k \mid k \in K_j(x)\} \subseteq V_B \cap X_j^-$, and all $K_j(x)$, $j=1,2,\ldots,j_x$, are disjoint. For each j, ASSIGN chooses the arcs in $OUT_j(x)$ from left to right, and, to each of the chosen arcs, assigns the commodities in the left-to-right order of their sinks t^k, $k \in K_j(x)$.

We shall show below after the description of ASSIGN that N is feasible if and only if ASSIGN succeeds, i.e., all arcs in G are assigned their flows within their capacities. In the following, f(a,k) denotes the flow value of commodity

k on arc a, and f_x^k denotes the flow value of commodity k which must go out of x. A node x is called scanned if flow assignment to all the arcs in OUT(x) has been completed.

Procedure ASSIGN

Input: A capacity balanced network $N=(G,P,g,c)$ satisfying Assumption A, B and C.

Output: Flow values $f(a,k)$ for all $a \in A$ and $k \in \{1,\ldots,K\}$ (if N is feasible), or an indication of infeasibility.

0. $f(a,k):= 0$, $f_x^k := 0$ for all $a \in A$, $x \in V$, $k \in \{1,\ldots,K\}$. All nodes are "unscanned". Compute $IN_j(x), OUT_j(x)$ and $K_j(x)$ $j=1,\ldots,j_x$, for all nodes x.

1. If all nodes are scanned, halt; a feasible flow assignment has been made. Otherwise take an unscanned node x in the minimum level. Let

$$f_x^k := \begin{cases} \sum_{a \in IN(x)} f(a,k) + g^k, & \text{if } x = s^k \\ \sum_{a \in IN(x)} f(a,k) - g^k, & \text{if } x = t^k \\ \sum_{a \in IN(x)} f(a,k), & \text{otherwise.} \end{cases}$$

If the following (T-1) or (T-2) hold, then halt by concluding infeasibility.

(T-1) $OUT(x)=\emptyset$ and $f_x^k > 0$ for some k.

(T-2) $f_x^k < 0$ for some k.

Otherwise, if $OUT(x)=\emptyset$ and $f_x^k = 0$ for all k, then let x be "scanned" and return to 1. Else if $OUT(x) \neq \emptyset$ and $f_x^k \geq 0$

for all k, then let

$K_j := K_j(x) \cap \{k \mid f_x^k > 0\}$, $j=1,2,\ldots,j_x$

$O_j := OUT_j(x)$, $j=1,2,..,j_x$,

and go to 2.

2. Repeat the following procedure for each $j=1,\ldots,j_x$ (in this order):

Until $O_j=\emptyset$ and $K_j=\emptyset$ holds, take the leftmost $a \in O_j$ and the leftmost commodity $k \in K_j(x)$ and let

$f(a,k) := c(a)$, $f_x^k := f_x^k - c(a)$, $O_j := O_j - \{a\}$ if $c(a) < f_x^k$,

$f(a,k) := f_x^k$, $c(a) := c(a) - f_x^k$, $K_j := K_j - \{k\}$ if $c(a) > f_x^k$,

$f(a,k) := c(a)$, $O_j := O_j - \{a\}$, $K_j := K_j - \{k\}$ if $c(a) = f_x^k$.

Whenever the following case (T-3) occurs, halt by concluding infeasibility.

(T-3) $K_j \neq \emptyset$ and $O_j=\emptyset$, or $O_j=\emptyset$ and $K_j=\emptyset$.

3. (It holds $O_j=K_j=\emptyset$ for $j=1,2,\ldots,j_x$.) If the following (T-4) holds, then halt by concluding infeasibility.

(T-4) $f(a(x,v),k)>0$ and $\Pi(v,t^k) \neq \emptyset$ for some $a(x,v) \in OUT(x)$ and k.

Otherwise let x be "scanned" and return to 1. □

Although ASSIGN described in [NAGA 87a] does not contain the condition (T-4) terminating the procedure, the

validity of (T-4) is obvious ((T-4) may terminate the procedure earlier).

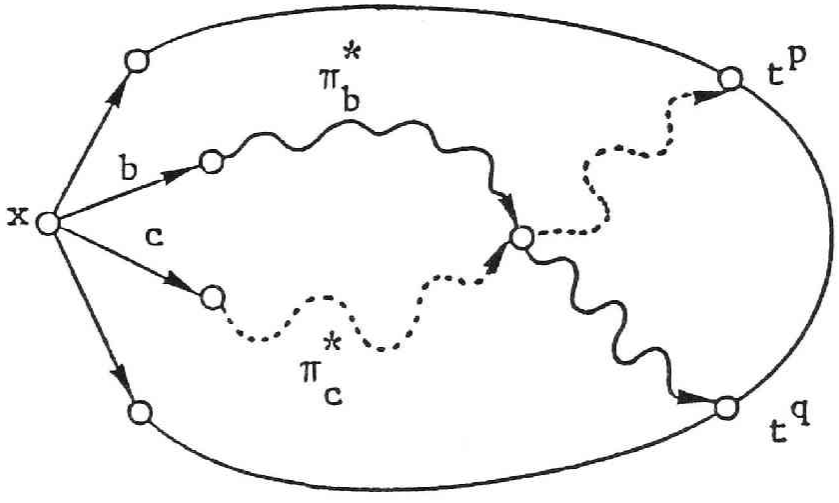


Fig. 3.1 Proof of Lemma 3.1.

3.4 CORRECTNESS OF ASSIGN

To prove the correctness of algorithm ASSIGN, we first prove the following Lemma 3.1. If N has more than one feasible solution, call the feasible solution $\{\tilde{f}(a,k) \mid a \in A,\ k=1,\ldots,K\}$ with the following property standard: If $\tilde{f}(a,k)>0$ for some arc $a \in OUT_j(x)$ and $k \in K_j(x)$, then $\tilde{f}(a',k')=0$ holds if $a' \in OUT_j(x)$ is to the right of arc a and $k \in K_j(x)$ is to the left of commodity k.

Lemma 3.1 If N=(G,P,c) is feasible, there exists a standard feasible solution.

Proof. We first consider a divergent node x. Clearly $j_x=1$ (i.e., $OUT_1(x)=OUT(x)$). Assume that a given feasible solution $\tilde{f}$ is not standard on x, i.e., there exist $b,c \in OUT_1(x)$ and $p,q \in K_1(x)$ such that b is to the left of c, p is to the left of q, $\tilde{f}(b,q)>0$ and $\tilde{f}(c,p)>0$ (see Figure 3.1). Since N is capacity-balanced, all arcs $a \in A$ are saturated, i.e.,

$$c(a)=\sum_{k=1}^{K} \tilde{f}(a,k).$$

Define

$$\Pi_b(x,t^q,\tilde{f}) \triangleq \{\pi \in \Pi(x,t^q) \mid b \in A(\pi),\ \tilde{f}(a,q)>0 \text{ for all } a \in A(\pi)\}$$

$$\Pi_c(x,t^p,\tilde{f})$$
$$\triangleq\{\pi\in\Pi(x,t^p)\,|\,c\in A(\pi),\ \tilde{f}(a,p)>0 \text{ for all } a\in A(\pi)\}.$$

Obviously these sets are not empty, and any $\pi_b\in\Pi_b(x,t^q,\tilde{f})$ and $\pi_c\in\Pi_c(x,t^p,\tilde{f})$ have a common node by the planarity of N. Let z be the common node nearest to x, and let π_b^* (π_c^*) be the part of π_b(π_c) from x to z. We assume that π_b and π_c are chosen so that the area surrounded by π_b^* and π_c^* does not contain in its interrior any node of a path in $\Pi_b(x,t^q,\tilde{f})\cup\Pi_c(x,t^p,\tilde{f})$. Then we modify the flows $\tilde{f}$ on π_b^* and π_c^* as follows:

$$\tilde{f}'(a,q):=\tilde{f}(a,q)-e,\ \tilde{f}'(a,p):=\tilde{f}(a,p)+e,\ \text{for } a\in A(\pi_b^*),$$
$$\tilde{f}'(a,p):=\tilde{f}(a,p)-e,\ \tilde{f}'(a,q):=\tilde{f}(a,q)+e,\ \text{for } a\in A(\pi_c^*),$$

where $e=\min[\min\{\tilde{f}(a,q)\,|\,a\in A(\pi_b^*)\},\min\{\tilde{f}(a,p)\,|\,a\in A(\pi_c^*)\}]$.

The resulting flow $\tilde{f}'$ is clearly feasible. Furthermore $\Pi_b(x,t^q,\tilde{f}')\cup\Pi_c(x,t^p,\tilde{f}')\subsetneq\Pi_b(x,t^q,\tilde{f})\cup\Pi_c(x,t^p,\tilde{f})$ holds, since at least one of $\tilde{f}'(a,q)$ ($a\in A(\pi_b^*)$) or $\tilde{f}'(a,p)$ ($a\in A(\pi_c^*)$) becomes 0 by the above modification, and no new path in $\Pi_b(x,t^q,\tilde{f}')\cup\Pi_c(x,t^p,\tilde{f}')$ is created as obvious from the assumption on π_b and π_c. Therefore after finite repetition of such modifications, we obtain a feasible solution $\tilde{f}$ for which either $\Pi_b(x,t^q,\tilde{f}')=\emptyset$ or $\Pi_c(x,t^p,\tilde{f}')=\emptyset$

holds.

Now if $\tilde{f}$ is not standard on OUT(x), we apply the above operation to the nonstandard pair of arcs b,c $\in$OUT(x), which is leftmost in the lexicographical sense of (b,c). Repeating this, we eventually obtain a feasible flow $\tilde{f}$ that is standard on x, because once a pair (b,c) becomes standard, it will never become nonstandard again, as easily shown.

This procedure is then applied to all nodes x in the increasing order of their levels. A nondivergent node x may have $j_x>1$, but it is obvious that the above procedure can be applied to $j=1,2,\ldots,j_x$ separately. Once $\tilde{f}$ becomes standard on x, it remains to be standard on x even if the above modification is applied to the nodes of larger levels. Therefore, we eventually obtain a feasible solution $\tilde{f}$ that is standard on all nodes. □

Lemma 3.2 If N=(G,P,g,c) is feasible, the standard feasible solution is unique.

Proof. Assume two distinct standard feasible solutions f and $\tilde{f}$, and take a node x such that $f(a,k) = \tilde{f}(a,k)$ for all k and $a \in OUT(y)$ with level(y)<level(x), but $f(a',k') \neq \tilde{f}(a',k')$ for some $a' \in OUT_j(x)$ and k'. Assume

without loss of generality that a' is the leftmost arc in $OUT_j(x)$ with this property, and that

$$0 \leq \tilde{f}(a',k') < f(a',k'). \tag{3.1}$$

To satisfy the capacity constraint of a', there exists a k" such that

$$0 \leq f(a',k") < \tilde{f}(a',k"). \tag{3.2}$$

(3.1) and (3.2) then imply that some arcs $b,c \in OUT_j(x)$ (possibly b=c), located to the right of a', satisfy

$$0 \leq f(b,k') < \tilde{f}(b,k') \tag{3.3}$$

$$0 \leq \tilde{f}(c,k") < f(c,k"). \tag{3.4}$$

If k' is to the left of k", (3.2) and (3.3) imply that $\tilde{f}$ is not standard, while if k" is to the left of k', (3.1) and (3.4) imply that f is not standard. In either case, this is a contradiction. □

Lemma 3.3 If a capacity balanced network N=(G,P,g,c) is feasible, ASSIGN finds the standard feasible solution. On the other hand, if N is infeasible, ASSIGN indicates the infeasibility by halting at (T-1), (T-2), (T-3) or (T-4).

Proof. If N is feasible, Lemma 3.1 says that it has the standard feasible solution. First, consider a node x with level(x)=0. As easily proved from the order of selections of arcs a and commodities k in Step 2 of ASSIGN, the flows given to the arcs in OUT(x) by ASSIGN are standard, and this is the only way to have standard flows on OUT(x) under the condition that all arcs in OUT(x) are saturated. Since we assume that N is feasible, and the standard feasible flows on OUT(x) are unique by Lemma 3.2, ASSIGN realizes on OUT(x) exactly the same flows as the standard feasible flows. To use induction, take a node x and assume that the flows realized by ASSIGN on OUT(y) of all y with level(y)<level(x) are the only ones that is both feasible and standard. These flows uniquely determine the flows on IN(x). Given the flows on IN(x), it is also easly to see that the flows on OUT(x) realized by ASSIGN are the only ones that are feasible and standard at x. This shows that ASSIGN gives the standard feasible flow if N is feasible.

Finally we consider the case in which N is infeasible. Assume that ASSIGN has scanned all nodes x in N successfully, then it is easily seen that the flow realized by ASSIGN is feasible since it satisfies the constraints of

flow conservation and capacity. This is a contradiction. The only way not to reach this conclusion is to halt at one of (T-1)-(T-4). □

3.5 TERMINATION CONDITION OF ASSIGN

As described in Lemma 3.3, ASSIGN halts only at (T-1)-(T-4) when an infeasible network is input. In this section, we show that (T-4) always occurs earlier than (T-1)-(T-3). That is, (T-4) in Step 3 is sufficient to terminate ASSIGN if a given network is infeasible.

For a node x and a node set $X \subseteq V$ define

$$REACH(x)=\{v \in V \mid \Pi(v,x) \neq \emptyset\} \cup \{x\},$$

$$OUT(X)= \sum_{x \in X} OUT(x)$$

Let SCAN(x) denote the set of nodes already scanned when an unscanned node x is chosen in Step 1 of ASSIGN (in particular $x \notin SCAN(x)$). At this instant, all arcs in OUT(SCAN(x)) are already assigned flows to their capacities. Since ASSIGN chooses each node x in the nondecreasing order of level(x),

$$REACH(x) \subseteq SCAN(x) \cup \{x\}$$

always holds.

For an infeasible CB network, ASSIGN halts by one of (T-1)-(T-4). We have the following properties.

Lemma 3.4 For an infeasible CB network satisfying Assumption B, ASSIGN does not halt by (T-1) or (T-2).

Proof. First assume that $OUT(x)=\emptyset$ but $f_x^k \neq 0$ for some k. Since this x is a convergent node, we have $x \notin S$ holds by Assumption, and

$$c(x) = \sum_{t^k=x} g^k - \sum_{a \in IN(x)} c(a) = 0$$

holds. As flow values in IN(x) have already been determined by ASSIGN, this means

$$\sum_k \sum_{a \in IN(x)} f(a,k) = \sum_{t^k=x} g^k.$$

and hence

$$\sum_k f_x^k = 0$$

as obvious from Step 1 of ASSIGN. Therefore, if $f_x^k \neq 0$ for some k, $f_x^{k'} < 0$, i.e., (T-2), holds for some k'.

Now we assume $f_x^k < 0$ for some k and derive a contradiction. By Step 1, $f_x^k < 0$ is possible only if $x=t^k$. Also we have $\Pi(s^k,t^k) \neq \emptyset$ by Assumption B and $s^k \in REACH(x)-\{x\} \subseteq SCAN(x)$. By the mechanism of ASSIGN, all flows in OUT(SCAN(x)) have already been determined. If $f(a(u,v),k)>0$ for some $a(u,v) \in A(REACH(x)-\{x\})$, this implies $\Pi(v,t^{k'})=\emptyset$, and ASSIGN must have halted in Step 3 by (T-4) when it has scanned node $u \in SCAN(x)$. Therefore commodity k from source s^k passes through only nodes in REACH(x) and reaches sink $t^k=x$. This and flow conservation imply $f_x^k=0$,

contradicting the assumption. □

For two nodes x and y such that $\Pi(x,y)\neq\emptyset$, we define two directed paths from x to y $\pi_R(x,y)$ and $\pi_L(x,y)$ as follows.

$\pi_R(x,y)$: Any arc $a(u,v)\in OUT_j(u)$ in $\pi_R(x,y)$ satisfies that if an arc $a(u,w)\in OUT_j(u)$ is to the right of $a(u,v)$, then $\Pi(w,y)=\emptyset$.

$\pi_L(x,y)$: Defined similarly as above by replacing "right" with "left".

Lemma 3.5 For an infeasible CB network N satisfying Assumption B, ASSIGN halts by (T-4).

Proof. If ASSIGN does not halt by (T-4), it terminate by (T-3) from Lemma 3.4. Assume (T-3) holds at $x=x^*$ in Step 2 of ASSIGN. From $\Delta c(x^*)=0$,

$$\sum_k f_x^k = \sum_{a\in OUT(x^*)} c(a) \qquad (3.5)$$

holds before Step 2 is performed. First we show $j_x\geq 2$ for $OUT_j(x)$, $j=1,2,\ldots,j_x$. Since Step 2 is performed only if $OUT(x^*)\neq\emptyset$, $j_{x^*}\geq 1$ clearly holds. Assume $j_{x^*}=1$. Then the condition $O_1=\emptyset$ and $H_1\neq\emptyset$ in (T-3) is impossible. Therefore assume $O_1\neq\emptyset$ and $K_1=\emptyset$. If $f_{x^*}^{k1}>0$ for some $k1\in K_1$ by $H_1\neq\emptyset$, then $f_{x^*}^{k2}<0$ for some k2 by $O_1\neq\emptyset$ and (3.5). This means that

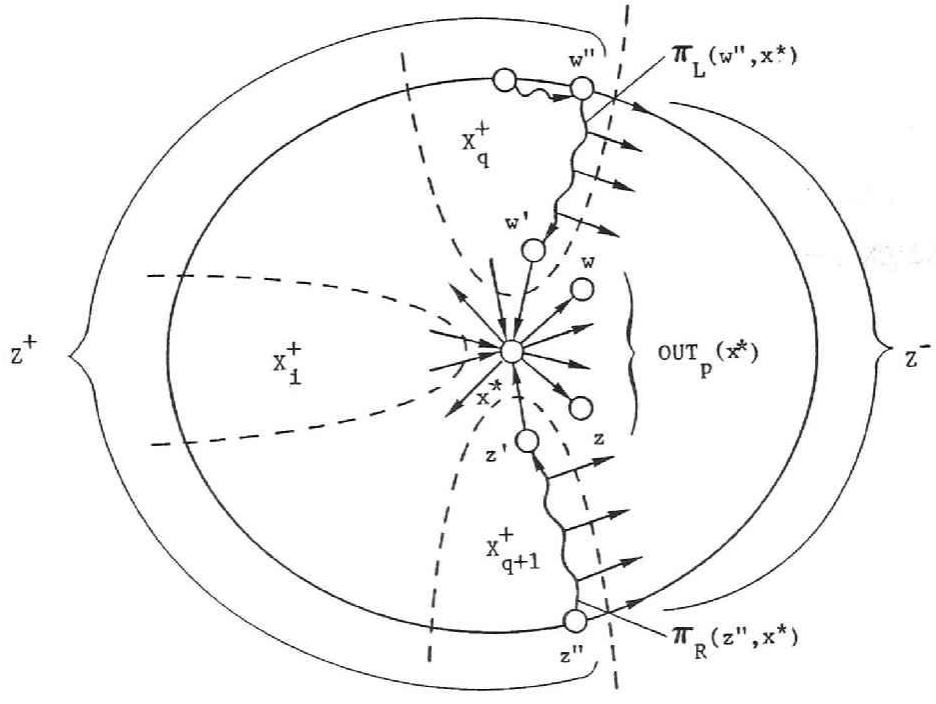


Fig. 3.2 Proof of Lemma 3.5.

ASSIGN has halted before choosing x^*, a contradiction. Therefore, assume $f^{k}_{x^*}=0$ for all $k \in K_1$. Then $f^{kl}_{x^*}>0$ for some $kl \notin K_1(x^*)$ by (3.5). As $\Pi(x^*,t^{kl})=\emptyset$, we have $x^* \neq s^{kl}$. At this instant, there exists a node v satisfying $a(v,x^*) \in IN(x^*)$ and $f(a(v,x^*),kl)>0$, i.e., (T-4) holds. Again ASSIGN must have halted before choosing x^*, a contradiction. Therefore we obtain $j_{x^*} \geq 2$.

To prove the lemma, assume that we continue Step 2 of ASSIGN for all $j=1,2,\ldots,j_{x^*}$ even if some j satisfying (T-3) are found. By flow conservation constrained for node x, we have some j satisfying $O_j=\emptyset$ and $K_j \neq \emptyset$ and other j satisfying $O_j \neq \emptyset$ and $K_j=\emptyset$. Assume $O_j=\emptyset$ and $K_j \neq \emptyset$ for j=p. See Figure 3.2. Since all flows in $OUT_p(x^*)$ have already been determined, $OUT_p(x^*)$ is saturated and $f^{kl}_{x^*}>0$ for $kl \in K_p(x^*)$. Here we show that source s^{kl} and sink t^{kl} are separated by an unilateral simple cut not satisfying the cut-condition. Let $a(x^*,w)$ and $a(x^*,z)$ be the leftmost and the rightmost arcs in $OUT_p(x^*)$, respectively. Further let $a(w',x^*)$ and $a(z',x^*)$ be the arcs next to $a(x^*,w)$ and $a(x^*,z)$ in $IN(x^*)$, respectively. (Similar argument holds even if $a(w',x^*)$ or $a(z',x^*)$ does not exist.) By Lemma 2.5, there are connected divergent sets X^+_i, $i=1,2,..,i_{x^*}$. Without loss of generality, assume

$a(w',x^*) \in IN_q(x^*)$ and $a(z',x^*) \in IN_{q+1}(x^*)$. Since set X_q^+ is divergent, it contains $REACH(x^*)$. Let $w'' \in REACH(w') \cap V_B$ be the rightmost node in the part of the boundary contained in X_q^+. Similarly, let $z'' \in REACH(z') \cap V_B$ be the leftmost node in the part of the boundary contained in X_{q+1}^+. We consider the undirected path C consisting of $\pi_L(w'',x^*)$ and $\pi_R(z'',x^*)$. C separates the graph into two parts. Let Z^+ denote the set consisting of the nodes in C and the nodes in the part (divided by C) not containing the end nodes of $OUT_p(x^*)$. Further let $Z^- = V - Z^+$. By the selection rule of w" and z", there is no directed path from a node in Z^- to a node in C, i.e., Z^+ is divergent. As each initial node of arc in $A(Z^+)$ belongs to path C, we have $|A(Z^+) \cap A_B| = 2$. Let Z_i^+, i=1,2,...,m, denote components of Z^+. By Lemma 2.4, $|A(Z_i^+) \cap A_B| \geq 2$ for i=1,2,...,m. Since no arc exists between Z_i^+ and $Z_{i'}^+$ ($i \neq i'$), i.e., $A(Z_i^+)$ and $A(Z_{i'}^+)$ are disjoint, it holds $|A(Z^+) \cap A_B| \geq 2m$. Therefore m=1 and Z^+ is connected. Similarly we may obtain from $|A(V-Z^-) \cap A_B| = |A(Z^+) \cap A_B| = 2$ that Z^- is connected.

Now we consider the cut $A(Z^+)$. This is a unilateral simple cut as obvious from the connectivity of Z^+ and Z^-. Since all initial nodes of arcs in $A(Z^+)$ are in path C,

they are scanned and flow is already assigned to each arc in $A(Z^+)$. Here we show, for any $a \in A(Z^+)$, that

$$f(a,k)>0 \text{ then } t^k \in Z^- \tag{3.6}$$

holds. In the case that $a \in OUT_p(x^*)$, we have a connected convergent set X_p^- by applying Lemma 2.5 to $OUT_p(x^*)$. X_p^- satisfies $\{t^k | k \in K_p(x^*)\} \subseteq X_p^- \cap V_B$ and further $X_p^- \subseteq Z^-$ from its convergency. Then $\{t^k | k \in K_p(x^*)\} \subseteq Z^-$ and (3.6) holds for $a \in OUT_p(x^*)$ by the assumption $O_p=\emptyset$. Take $a(u,v) \in A(Z^+)-OUT_p(x^*)$, and assume $f(a(u,v),k)>0$ with $t^k \notin Z^-$. Then $\Pi(v,t^k)=\emptyset$ holds from $v \in Z^-, t^k \notin Z^-$ and convergency of Z^-, and this means that ASSIGN halts by (T-4) when node u is chosen, contradicting that u is scanned. Therefore we obtain (3.6).

Now we show that $A(Z^+)$ separates (s^{kl},t^{kl}) and does not satisfy the cut-condition. Since $A(Z^+)$ is saturated, we have

$$\sum_k \sum_{a \in A(Z^+)} f(a,k) = c(a). \tag{3.7}$$

At this instant, flow $f_{x^*}^{kl}$ is not yet assigned to any arc from node x^*. Since no flow passes through a unilateral cut twice, we have the following from (3.6) and the definition of $f_{x^*}^{kl}$.

$$\sum_k \sum_{a \in A(Z^+)} f(a,k) + f_{x^*}^{kl} \leq g(Z^+). \tag{3.8}$$

By (3.7) and (3.8),

$$r(Z^+)=c(Z^+)-g(Z^+)\leq -f_{x^*}^{kl}<0.$$

This shows that a unilateral simple cut $A(Z^+)$ does not satisfy the cut-condition under the assumption that ASSIGN halts by (T-3). This, however, contradicts Lemma 2.8, and the lemma is proved. □

From the above argument, we see that (T-4) in Step 3 is sufficient to terminate procedure ASSIGN if a given network is infeasible.

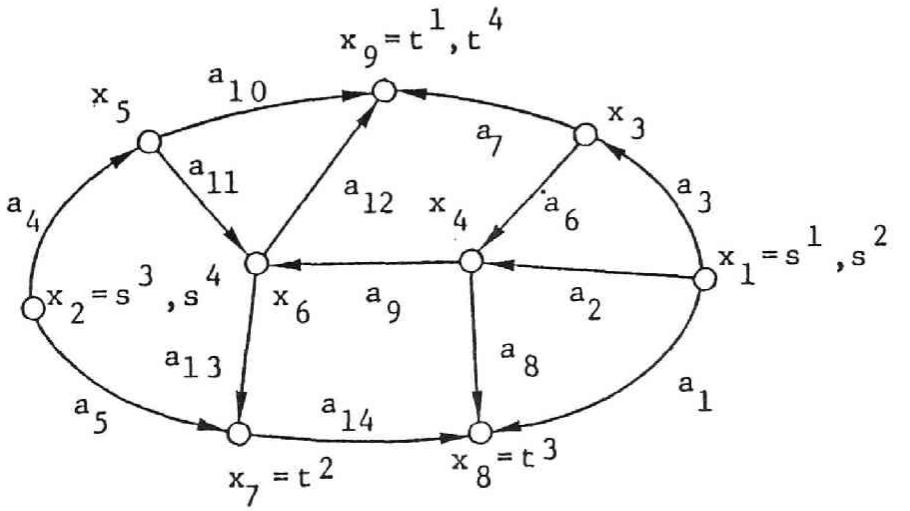
3.6 COMPUTATIONAL COMPLEXITY OF ASSIGN

In ASSIGN, all major operations are additions or subtractions. This implies that the integral property holds for the standard feasible flow realized by ASSIGN, i.e., flows f(a,k) are always integer if all capacities c(a) are nonnegative integers.

The preprocessing to check whether a CB network satisfies Assumption B or not requires $O(K|A|)$ time. The computation of $K_j(x)$ in Step 0 of ASSIGN is done by tracing all nodes in the decreasing order of levels (i.e., from sinks to sources). This is done in $O(K|A|)$ time. The number of operations required in Steps 1 and 2 of ASSIGN for each node x is $O(|OUT(x)|+K)$. Since $\sum_{x \in V} |OUT(x)|=|A|$, ASSIGN requires $O(K|A|+K|V|)$ time in total. Furthermore $O(|A|) = O(|V|)$ holds by the planarity of N, and the time becomes $O(K|V|)$. The required memory space is $O(K|V|)$ for storing the flows in arcs. Consequently we obtain the next theorem.

Theorem 3.1 The integral flow property holds for a capacity blanced network N=(G=(V,E),P,g,c) satisfying Assumption B. Procedure ASSIGN decides whether N is feasible or not, and gives the standard feasible flow if N is feasible. The time

and space required by ASSIGN are $O(K|V|)$. □



$g^1 = 20, g^2 = g^3 = g^4 = 10$

capacity: $c(a_i) = 20 (i = 3, 6, 9, 13)$

$c(a_i) = 10 (i = 1, 2, 4, 5, 7, 8, 10, 12, 14)$

Fig. 3.3 Example of a CB network.

Example 3.1 In Figure 3.3 and Table 3.1, an example of CB network with K=7 and its numerical results are given. ASSIGN scans the nodes in the order of $x_1, x_2, \ldots, x_9$ and assigns flows to arcs in the order of $a_1, a_2, \ldots, a_{14}$. The values $f(a,k), a \in A, k=1,2,\ldots,K$, in Table 3.1 are the standard feasible flows obtained by ASSIGN and f_x^k is the values observed immediately after Step 1 of ASSIGN. In Table 3.1, commodities in { } of $K_j(x)$ are ordered from left to right. □

To compare the computation speeds of the simplex method for general linear programming and algorithm ASSIGN, [ITO 85] solved randomly generated CB networks. The size of problems, $L=K|A|$, varies from 30 to 60. The computational results confirm that ASSIGN works much faster than the simplex method. The speed of ASSIGN is roughly $0.009*L^2$ times faster than the simplex method.

Talbe 3.1 Values of $K_j(x)$, f_x^k and f(a,k).

x	$K_j(x)$
x_1, x_3, x_4	{3,6,7,2,5,1,4,}
x_2, x_5	{1,4,3,6,7,2,5}
x_6	$K_1(x_6)=\{1,4\}, K_2(x_6)=\{3,6,7,2,5\}$
x_7	{3,6,7}
x_8, x_9	ϕ

Values of f_x^k

$k \backslash x_i$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
1	20		20	10		10			
2	10			10		10			
3		10					10		
4		10			10				
5					10	10			
6			10	10					
7	10								

Values of $f(a,k)$

$k \backslash a_i$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}
1			20			10	10		10			10		
2		10							10				10	
3					10									10
4				10						10				
5											10		10	
6						10		10						
7	10													

3.7 CONCLUSION

In this chapter, we introduced the concept of capacity balance. Based on it, we defined class CB of multicommodity flow problems for class CB in directed networks, and presented an efficient graph theoretic algorithm for it. In the following chapter, we will discuss an extension of class CB to a class of networks with capacity unbalanced nodes.

CHAPTER 4

ALGORITHM MATE1 FOR TESTING FEASIBILITY OF A CS NETWORK

4.1 INTRODUCTION

In this chapter, we introduce class CS (capacity semi-balanced networks), an extension of CB, and show that a CS network can be reduced to a CB network. Therefore, CS also has a polynomial time graph theoretic algorithm and the integral flow property [NAGA 87a]. This class contains certain multi-item multi-stage production scheduling problem [IBAR 82] as a special case, indicating its importance in practical applications.

4.2 DEFINITION OF DUMMY FLOW

When there is some node x with $\Delta c(x) \neq 0$, algorithm ASSIGN does not work correctly. To handle such nodes, we first remove all capacity unbalanced nodes by supplying a flow of the new commodity d to each unbalanced node, the amount of which is equal to the residue of capacity. This defines the multicommodity flow problem with K+1 commodoties (i.e., k=1,...,K,d) by adding sources and sinks of commodity d to the capacity unbalanced nodes in N.

Definition 4.1 For a network N, we call the following flows of new commodity d, dummy flows. If $\Delta c(x) > 0$, then we put a dummy source s_x^d at node x (i.e., $s_x^d = x$), and let $g_x^d = \Delta c(x) > 0$ denote the supply amount of s_x^d. Similarly if $\Delta c(x) < 0$, then we put a dummy sink t_x^d at node x, and let $g_x^d = -\Delta c(x) > 0$. Let

g^d: the set of g_x^d.

S^d: the set of all s_x^d 's. T^d: the set of all t_x^d 's

$P^d := (S^d, T^d)$, $N^d := (G, P, P^d, g, g^d, c)$.

The flows of d in N also satisfy constraints (1.5) and (1.6). □

Lemma 4.1 The multicommodity flow problem in N^d with commodities k=1,...,K,d is feasible if and only if the original problem in N with commodities k=1,...,K is feasible.

Proof. Obvious from (1.5),(1.6) and Definition 4.1. □

It is obvious that the capacity constraint is satisfied only if

$$\sum_{k=1,\ldots,K,d} f(a,k) = c(a)$$

holds. In other words, $\Delta c(x) = 0$ holds for all $x \in V$ in

the resulting network N^d. Note that commmodity d may have more than one source and/or one sink, and (3) of Assumption A does not generally hold.

If a feasible solution f to N^d has a positive e such that $f(a,d) \geq e$ (>0), $a \in A(\pi)$, for some $\pi \in \Pi(s_x^d, t_y^d)$, then this flow from s_x^d to t_y^d may be regarded as the flow of a new K+1-th commodity. If all dummy flows can be decomposed into new commodities by repeating this operation, then the resulting network becomes CB. The feasibility of the resulting network may be tested by ASSIGN. As such a reducible class, we consider the following CS (capacity semibalanced networks).

4.3 CS NETWORKS

Definition 4.2 We call that a multicommodity flow problem $N=(G,P,g,c)$ belongs to class CS if N satisfies the following conditions.

(1) N satisfies (1)-(3) of Assumption A.

(2) Let $S^d=\{x \mid \Delta c(x)>0\}$, $T^d=\{x \mid \Delta c(x)<0\}$. Then V can be divided into two connected sets X and V-X such that $S^d \subseteq X \cap V_B$ and $T^d \subseteq (V-X) \cap V_B$ (see Figure 4.1). □

As an example of CS, we present a certain multi-item multi-stage production scheduling problem in Figure 4.2. In this network, a slanting arc that advances one stage in each period represents the production in the corresponding stage. Such an arc is given a capacity with finite value. A vertical arc that advances one period represents the inventory and its capacity is considered to be infinite. From the nodes marked with "s", materials of item k are supplied, and finished items k are shipped from the nodes marked with "t". Since the source-sink pair of each product is specified, this may be reguarded as a multicommodity flow problem. This network does not satisfy the condition of capacity balance. We can, nevertheless, make all node except nodes marked "s" or "t" balanced without loss of

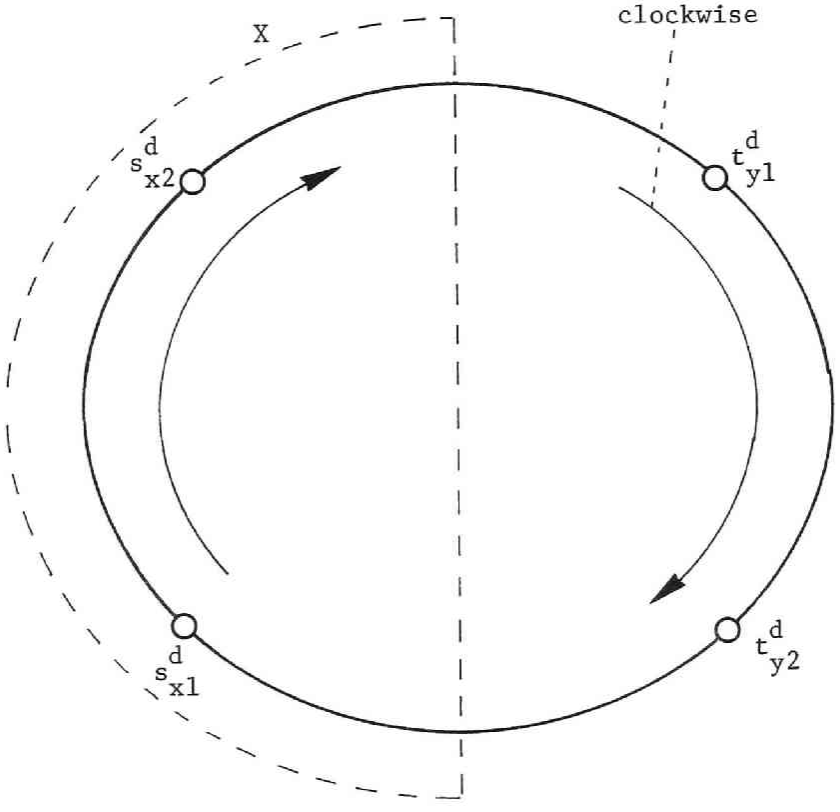


Fig. 4.1 Illustration of a CS network.

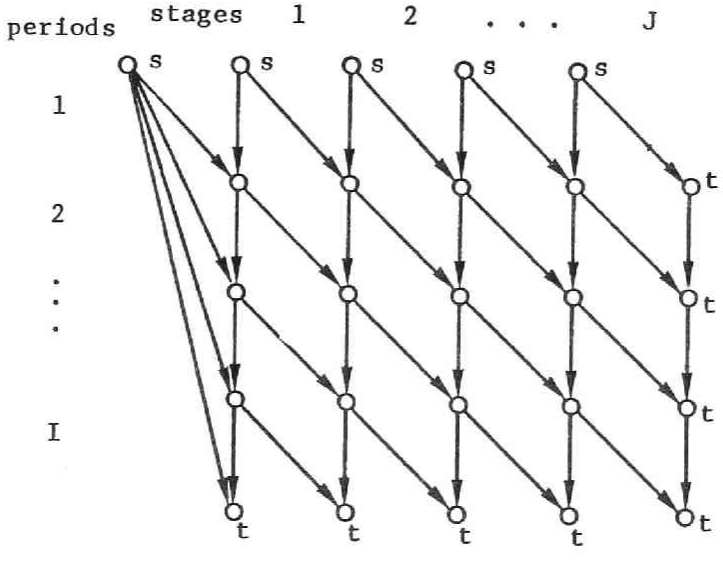


Fig. 4.2 Multi-item multi-stage production scheduling problem.

feasibility, by replacing (infinite valued) capacities of all inventory arcs with some capacities of sufficiently large finite values. As a result, a node x with $\Delta c(x) \neq 0$ exists only on the boundary and it is easily shown that a node x with $\Delta c(x) > 0$ ($\Delta c(x) < 0$) must be a node marked "s" ("t"), i.e., (2) of Definition 4.2 holds. Thus the network in Figure 4.2 may be considered as CS.

In order to construct an algorithm for solving a CS network, we shall derive some properties of a network in class CS.

From the definition of $\Delta c(x)$, any CS network clearly satisfies

$$\sum_{x \in S^d} g_x^d - \sum_{x \in T^d} g_x^d = 0.$$

We define the left-right relation between nodes in S^d (T^d) on the basis of (2) of Definition 4.2 as follows: For the set X in (2) of Definition 4.2, s_{x1}^d is to the left of s_{x2}^d if we visit s_{x1}^d before s_{x2}^d when we go round the boundary of the part contained in $X \cap V_B$ in the clockwise manner. See Figure 4.1. Similarly we define for T^d by using $(V-X) \cap V_B$. In Figure 4.1, t_{y1}^d is to the left of t_{y2}^d.

Lemma 4.2 Let N^d be the network constructed from a CS network by Definition 4.2. If N^d is feasible, the

rightmost node (say s_x^d) in S^d and the leftmost node (say t_y^d) satisfy the following: $\Pi(s_x^d,t_y^d)\neq\emptyset$ and there exist $\pi_1,\pi_2,\ldots,\pi_m \in \Pi(s_x^d,t_y^d)$ and e_i (i=1,2,...,m) such that

$$\sum_{i=1}^{m} e_i = \min\{g_x^d, g_y^d\}, \text{ for each arc } a \in A(\pi_i),$$

$$\sum_{i \mid a\, A(\pi_i)} e_i \leq f(a,d)$$

(summation runs over all i satisfying $a \in A(\pi_i)$)

Proof. For any feasiblbe solution f to N^d, let

$$\Pi^d(x,y) \triangleq \{\pi\in\Pi(x,y) \mid f(a,d)>0 \text{ for } a\in A(\pi)\}.$$

For each s_x^d and t_y^d, there exist some $t_v^d\in T^d$ and $s_w^d \in S^d$ satisfying $\Pi(s_x^d,t_v^d)\neq\emptyset$ and $\Pi(s_w^d,t_y^d)\neq\emptyset$. First, $t_v^d=t_y^d$ or $s_w^d=s_x^d$ implies $\Pi^d(s_x^d,t_y^d)\neq\emptyset$. On the other hand, if $t_v^d\neq t_y^d$ and $s_w^d\neq s_x^d$, then since t_v^d is to the right of t_y^d and s_w^d is to the left of s_x^d, any $\pi_a \in \Pi^d(s_x^d,t_v^d)\neq\emptyset$ and $\pi_b \in \Pi^d(s_w^d,t_y^d)\neq\emptyset$ have a common node z by the planarity of G. Thus, from $\Pi(s_x^d,z)\neq\emptyset$ and $\Pi(z,t_y^d)\neq\emptyset$, $\Pi^d(s_x^d,t_y^d)\neq\emptyset$ follows.

Here we choose a $\pi_1 \in \Pi^d(s_x^d,t_y^d)$ and let $e_1=\min\{f(a,d) \mid a\in A(\pi_1)\}$. Clearly $e_1\leq\min\{g_x^d,g_y^d\}$. If $e_1=\min\{g_x^d,g_y^d\}$ then the lemma is shown. If $e_1<\min\{g_x^d,g_y^d\}$ then we consider the network $\tilde{N}^d=(G,P,P^d,g,\tilde{g}^d,\tilde{c})$ defined by

$$\tilde{c}(a):=c(a)-e_1, \qquad a \in A(\pi_1)$$

$$\tilde{c}(a):=c(a), \qquad a \notin A(\pi_1)$$

$$\tilde{f}(a,d):=f(a,d)-e_1, \quad a \in A(\pi_1)$$

$$\tilde{f}(a,d):=f(a,d), \qquad a \notin A(\pi_1)$$

$$\tilde{g}_x^d:=g_x^d-e_1,\ \tilde{g}_y^d:=g_y^d-e_1,$$

$$\tilde{g}_w^d:=g_w^d \quad w\neq x,y.$$

Obviously $\tilde{f}$ is feasible to $\tilde{N}^d$. Then for

$$\tilde{\Pi}^d(x,y)\triangleq\{\pi\in\Pi(x,y)\,|\,\tilde{f}(a,d)>0 \text{ for } a\in A(\pi)\},$$

we get $\tilde{\Pi}^d(s_x^d,t_y^d)\neq\emptyset$ from $\min\{\tilde{g}_x^d,\tilde{g}_y^d\}>0$ in the same manner as above. Now we choose a $\pi_2\in\tilde{\Pi}^d(s_x^d,t_y^d)$ and repeat the same operation. As long as $\min\{\tilde{g}_x^d,\tilde{g}_y^d\}>0$, this repetition can be continued. Since $\min\{\tilde{g}_x^d,\tilde{g}_y^d\}$ must be reduced at least by 1 at each iteration. $\tilde{g}_x^d$ or $\tilde{g}_y^d$ must becomes 0 after $m(<\infty)$ iterations, and we obtain $\pi_1,\pi_2,\ldots,\pi_m$ and $e_1,e_2,\ldots,e_m$ satisfying the lemma's statement. □

4.4 ALGORITHM MATE1

For a CS network $N^d=(G,P,P^d,g,g^d,c)$ with commodities $k=1,2,\ldots,K$ and d, Lemma 4.2 asserts that N^d can be reduced to a CB network by applying the following procedure NEW-ITEM(s_x^d,t_y^d) to the the rightmost node s_x^d in S^d and the leftmost node t_y^d in T^d.

Procedure NEW-ITEM(s_x^d, t_y^d)

$e:= \min\{s_x^d,\ t_y^d\}$;

$s^{K+1}:= s_x^d$; $t^{K+1}:= t_y^d$; $g^{K+1}:= e$;

$S:= S \cup \{s^{K+1}\}$; $T:= T \cup \{t^{K+1}\}$;

$P:= P \cup (s^{K+1},\ t^{K+1})$;

$g_x^d:= g_x^d - e$; $g_y^d:= g_y^d - e$;

$S^d:= S^d - \{s_x{}^d\}$, if $g_x^d = 0$;

$T^d:= T^d - \{t_y{}^d\}$, if $g_y^d = 0$. □

By repeating this procedure, a CS network can be reduced to a CB network.

Procedure MATE1

Input: A CS network $N=(G,P,g,c)$ satisfying Definition 4.2, where $g=\{g^1,g^2,\ldots,g^K\}$.

Output: A CB network $\tilde{N}=(G,\tilde{P},\tilde{g},c)$ that is feasible if and only if so is N, where $\tilde{g}=\{g^1,g^2,\ldots,g^K,\ldots,g^{\tilde{K}}\}$.

0. Construct the network $N^d=(G,P,P^d,g,g^d,c)$ with commodities $k=1,2,\ldots,K,d$ according to Definition 4.1.
1. If $P^d=\emptyset$, then let $\tilde{P}:=P$ $\tilde{K}:=K$ and halt.
2. Apply procedure NEW-ITEM(s_x^d,t_y^d) to the rightmost node s_x^d in S^d and the leftmost node t_y^d in T^d. Let $K:=K+1$ and return to 1. □

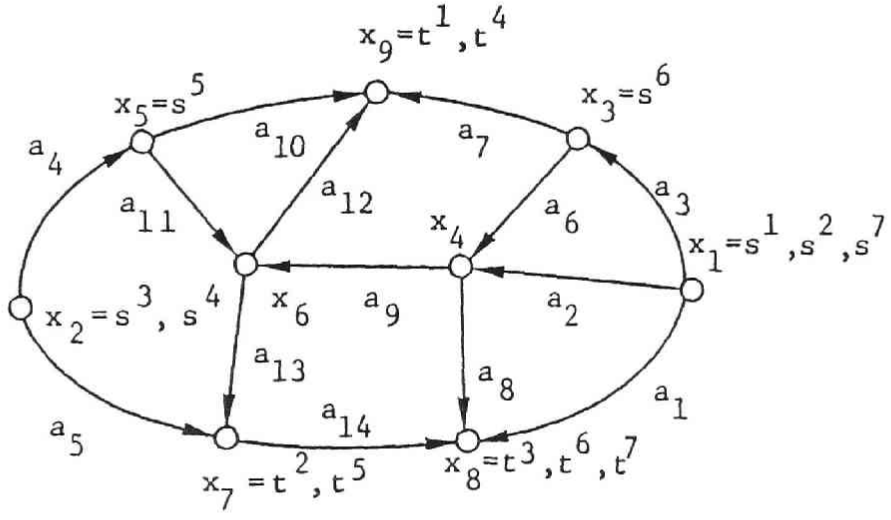
Since iterations of Step 2 in MATE1 is at most $|S^d|+|T^d|(\leq|V_B|)$, the required time of MATE1 is $O(|V_B|)$. The number of commodities generated by MATE1 is also $O(|V_B|)$.

Theorem 4.1 If a CS network $N=(G,P,g,c)$ satisfying Definition 4.2 is feasible, then N has a feasible flow with integral property. Procedure MATE1 and ASSIGN test its feasibility correctly. The time and space required by MATE1 and ASSIGN are $O((K+|V_B|)|V|)$.

Proof. To show the theorem, reduce N to a CB network by MATE1 and apply ASSIGN. Because this calculation needs only addition and subtraction, the integral flow property holds.

The resulting CB network with $O(K+|V_B|)$ commodities clearly shows the required time and space of the theorem. □

Example 4.1 We present a CS network in Figure 4.3. Here, $S^d=\{x_1,x_2,x_3\}$, $T^d=\{x_7,x_8\}$. Applying MATE1 to this network, dummy sources s_{x1}^d, s_{x3}^d, s_{x5}^d and dummy sinks t_{x7}^d, t_{x8}^d are first created by Definition 4.1 (where $g_{x1}^d=g_{x3}^d=g_{x5}^d=g_{x7}^d=10$, $g_{x8}^d=20$), and P^d is decomposed into source-sink pairs $(x_5,x_7),(x_3,x_8),(x_1,x_8)$ of new commodities. This completes the reducton to CB network. The obtained CB network is shown in Figure 3.3, and its feasibility can be tested by ASSIGN. □



$g^1=20, g^k=10 (k=2,3,4,5,6,7)$
capacity: $c(a_i)=20 (i=3,6,9,13)$
$c(a_i)=10 (i=1,2,4,5,7,8,10,11,12,14)$

Fig. 4.3 Example of a CS network.

4.5 CONCLUSION

In this chapter, by introducing the concept of dummy flows, we showed that a network in class CS can be reduced to a network in class CB, and develpoed an efficient graph theoretic algorithm for CS. Another class that is also reducible to class CB by a similar method will be discussed in the next chapter.

CHAPTER 5

ALGORITHM MATE2 FOR TESTING FEASIBILITY OF A CU NETWORK

5.1 INTRODUCTION

In this chapter, we introduce class CU (capacity semi-balanced unilateral networks) as another class reducible to CB, and develop $O(|V|^3)$ time algorithm that reduces CU to CB. In the next chapter, it will be shown that the max-flow min-cut theorem holds for class CB and CS, but CU does not have this property. In this sence, class CU is theoretically interesting because it can still be solved graph theoretically in polynomial time [NAGA 87b]. The multi-item multi-stage production scheduling problem [IBAR 82] is also a special case of class CU, and it indicates its importance in practical applications.

In the subsequent discussion, we solve two examples of CU network N_A and N_B, illustrated in Figure 5.1 and 5.2, respectively. The network N_B is presented by Kennington [KENN 78] as a counterexample to the max-flow min-cut theorem.

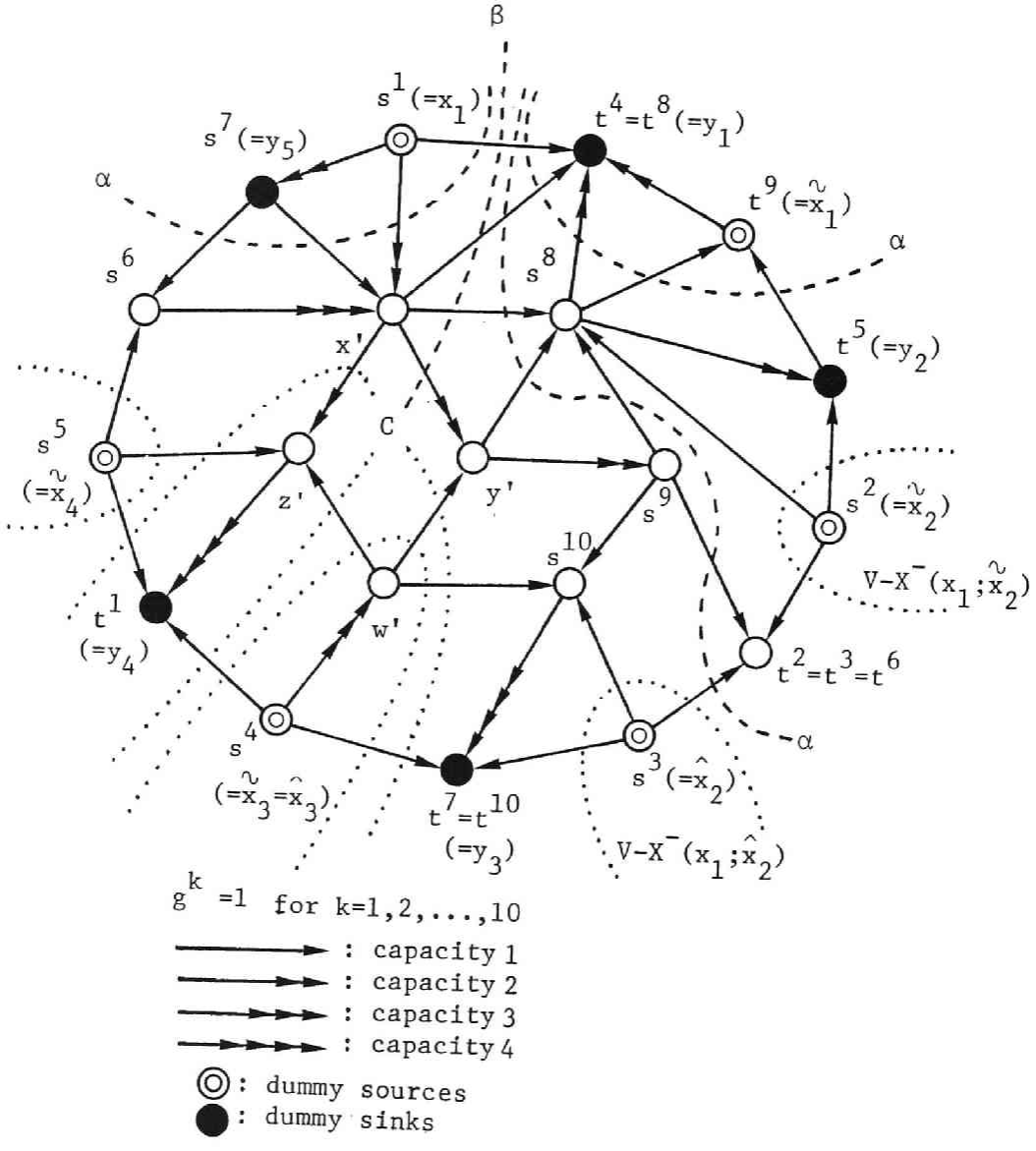


Fig. 5.1 CU network N_A.

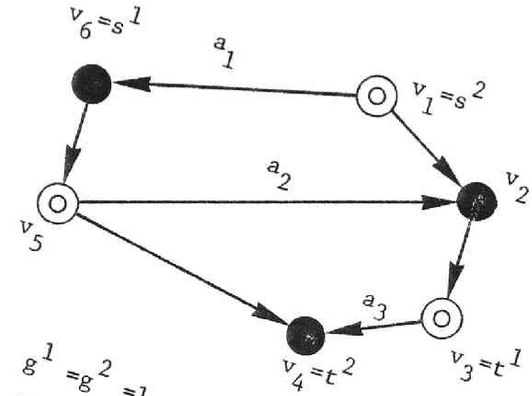


Fig. 5.2 CU network N_B.

5.2 CU NETWORKS

We define the unilaterality of a node in a network satisfying Assumption A as follows. Consider $IN_i(x)$, $i=1,2,\ldots,i_x$, and $OUT_j(x)$, $j=1,2,\ldots,j_x$ defined in Chapter 2. Here, a node x is called unilateral if $i_x \leq 1$ and $j_x \leq 1$, and semi-unilateral if $i_x * j_x = 2$. Note that $i_x \neq j_x$ occurs only for node x in the boundary. In Figure 5.1, node s^6 is semi-unilateral and other nodes are all unilateral. Further, if arcs in $OUT_j(x)$ are $a_1, a_2, \ldots, a_m$ (aranged in the clockwise manner), we say that a_i is to the left of a_j for $i<j$.

Definition 5.1 Multicommodity flow problem $N=(G,P,g,c)$ belongs to class CU if N satisfies the following conditions.

(1) N satisfies Assumption A.

(2) All node in G are unilateral or semi-unilateral.

(3) Let $S^d=\{x \mid \Delta\ c(x)>0\}$, $T^d=\{x \mid \Delta\ c(x)<0\}$. Then all nodes in $S^d \cup T^d \subseteq V_B$ are unilateral.

(4) $S^d \cup T^d \subseteq V_B$. □

For example, N_A in Figure 5.1 and N_B in Figure 5.2 are CU networks.

To test the feasiblity of a CU network and to find a

feasible flow if the network is feasible, we try to reduce a CU network to a CB network. For this purpose, we introduce dummy flow according to Definition 4.1. For N_A in Figure 5.1, we get $S^d=\{s^1,t^9,s^2,s^3,s^4,s^5\}$, $T^d=\{t^4,t^5,t^7,t^1,s^7\}$. In Figure 5.1, ◎ and ● repersent a dummy source and a dummy sink, respectively.

By Lemma 4.1, the multicommodity flow problem in $N^d=(G,P,P^d,g,g^d,c)$, with commodities $k=1,\ldots,K,d$, is feasible if and only if the original problem in $N=(G,P,g,c)$ with commodities $k=1,\ldots,K$ is feasible.

Note that commmodity d may have more than one source and/or one sink, and hence (3) of Assumption A does not generally hold.

If a feasible solution f to N^d has a positive e such that $f(a,d)\geq e(>0)$, $a\in A(\pi)$, for some $\pi\in\Pi(s_x^d,t_y^d)$, then this flow from s_x^d to t_y^d may be regarded as the flow of a new K+1-th commodity. If all dummy flows can be decomposed to new commodities by repeating this operation, then the resulting network becomes CB. The feasibility of the resulting network may be tested by ASSIGN. It will be shown that CU (capacity semi-balanced unilateral networks) is also such a reducible class.

Here, we introduce a unilateral simple cut. For nodes $x,y \in V$ with $\Pi(x,y)=\emptyset$, let X' be the set of nodes reachable from x. Clearly X' may be obtained by computing a maximal directed tree with root x. Time required for this computaton is $O(|A|)$. V-X' is composed of connected divergent sets $X_k, k=1,2,\ldots,m$. No arcs exist between distinct sets X_k, and some arcs from X_k to X' exist between X_k and X'. Let $X_{k'}$ denote the X_k including y, and $X^-(x;y)$ denote the connected convergent set $V-X_{k'}$. Similarly let X'' be the set of nodes reachable to y, where $\Pi(x,y)=\emptyset$. Among the components in V-X'', let X''_k be the connected convergent set containing y, and $X^+(y;x)$ denote $V-X''_k$.

For example, if $x=s^1, y=s^4$ in Figure 5.1, then we get

$X'=\{s^1,t^4,t^9,t^5,t^2,t^7,t^1,s^6,s^7,s^8,s^9,s^{10},x',y',z'\}$,

$X_1=\{s^2\}$, $X_2=\{s^3\}$, $X_3=\{s^4,w'\}$, $X_4=\{s^5\}$, $X^-(s^1;s^4)=V-X_3$.

If $y=t^7, x=s^2$ in Figure 5.1, then we have

$X''=\{t^7,s^3,s^4,s^5,s^6,s^7,s^1,x',y',w',s^9,s^{10}\}$,

$X''_1=\{t^4,t^9,t^5,s^2,t^2,s^8\}$, $X''_2=\{t^1,z'\}$

$X^+(t^7;s^2)=V-X''_1$.

Lemma 5.1 If a directed graph is connected (neglecting the directions of arcs), then for any pair of nodes $x,y \in V$ with $\Pi(x,y)=\emptyset$, the above sets $X^-(x;y)$ and $X^+(y;x)$ satisfy the

following.

$\Pi(x,v) \neq \emptyset$ for all nodes v with $a(u,v) \in A(V-X^{-}(x;y))$.

$\Pi(u,y) \neq \emptyset$ for all nodes u with $a(u,v) \in A(X^{+}(y;x))$.

Furthermore, $A(V-X^{-}(x;y))$ and $A(X^{+}(y;x))$ are unilateral simple cuts respectively.

Proof. Obvious from the definition of $X^{+}(y;x)$ and $X^{-}(x;y)$. □

In Sections 5.3-5.5, we clarify the properties of CU. Then in Section 5.6, we show that CU can be reduced to CB by algorithm MATE2.

5.3 A POLE OF DUMMY FLOWS

For two sets of nodes X and Y with $X \cap Y = \emptyset$, we define

$$d(X) = \sum_{s_x^d \in X} g_x^d - \sum_{t_y^d \in Y} g_y^d. \qquad (5.1)$$

If a network is feasible, the cut-condition:

$$r(X) \geq 0 \qquad \text{for all } X \subseteq V$$

clearly holds. Furthermore, for any connected divergent set X, the following condition holds.

$$d(X) \geq 0$$

A <u>boundary line</u> refers to an undirected path representing a part of the boundary. A boundary line from node x to node y refers to the undirected path when we go around the boundary from x to y in the clockwise manner. Note that $S^d \cup T^d \subseteq V_B$ holds by Definition 5.1. Then N^d with $S^d \cup T^d \neq \emptyset$ has two nodes $x = s_x^d \in S^d$ and $y = t_y^d \in T^d$, such that no other dummy sources and sinks exist in the boundary line from x to y (or y to x). We call such (s_x^d, t_y^d) a <u>neighbouring pair</u>. If a neighbouring pair (x,y) satisfies $\Pi(x,y) \neq \emptyset$, then (x,y) is called a <u>pole</u>. In Figure 5.1, (s^1, s^7) is a pole, and (s^5, s^7) is a neighbouring pair (but not a pole).

Lemma 5.2 If G of $N^d=(G,P,P^d,g,g^d,c)$ is a connected (neglecting the directions of arcs) planar directed graph and $\emptyset \neq S^d \cup T^d \subseteq V_B$, then a feasible N^d has at least one pole.

Proof. Assume that N^d has no pole. From $|S^d| \geq 1$ and $|T^d| \geq 1$, there exists a neighbouring pair (x_1,y_1) with $x_1 \in S^d$, $y_1 \in T^d$. Without loss of generality, we assume that there are no dummy sources and sinks in the boundary line from x to y. Since $\Pi(x_1,y_1)=\emptyset$ from the assumption, $V_1^*=X^+(y_1;x_1)$ exists by Lemma 5.1, where $y_1 \in V_1^*$ and $x_1 \notin V_1^*$. From the feasibility of N^d, $d(V_1^*) \geq 0$ holds, and this means $V_1^* \cap S^d \neq \emptyset$. Here, let x_2 be the node in $V_1^* \cap S^d$ nearest to y_1 in the boundary line. If there exsits another dummy sink in the boundary line from y_1 to x_2, let y_2 be the dummy sink nearest to x_2. Ohterwise, let $y_2=y_1$. See Figure 5.3. Here (x_2,y_2) is a neighbouring pair, and $\Pi(x_2,y_2)=\emptyset$ holds by assumption. Then $V_2^*=X^-(x_2;y_2)$ exists by Lemma 5.1, where $x_2 \in V_2^*$, $y_2 \notin V_2^*$. Now we show $V_2^* \subsetneq V_1^*$ as follows. Assume $V_2^*-V_1^* \neq \emptyset$, then there exists an arc $a(u,v) \in A(V_1^*)$ satisfying $u \in V_1^* \cap V_2^*$ and $v \in V_1^* - V_2^*$ from the divergency of V_1^*. Apply Lemma 5.1 to V_1^*, and we have $\Pi(u,y_1) \neq \emptyset$ for the above node u. This, however, contradicts the convergency of V_2^* by $u \in V_2^*$. Then we get $V_2^* - V_1^* = \emptyset$, and $V_2^* \subsetneq V_1^*$ holds from $y_2 \in V_1^*$, $y_2 \notin V_2^*$. We may define the neighbouring pair

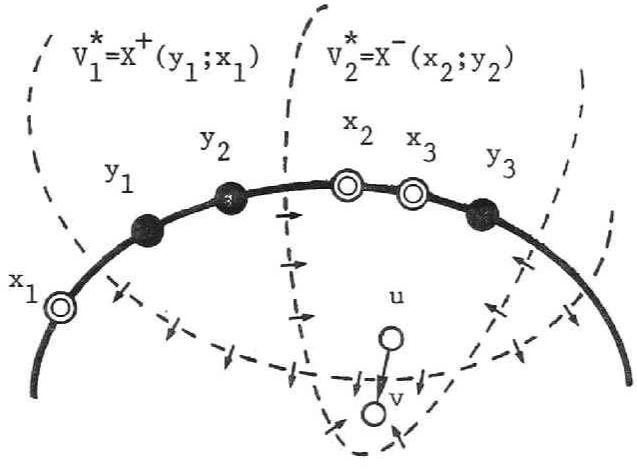


Fig. 5.3 Proof of Lemma 5.2.

(x_i', y_i) in the same way. This argument cannot be repeated indefinitely, however, because V is finite and $V_{i+1}^* \subsetneq V_i^*$ always holds. This proves the lemma. □

Since the proof of Lemma 5.2 does not need (2) of Assumption A, we obtain the following corollary.

Corollary 5.1 Even if G has divergent and/or convergent nodes outside of the boundary in Lemma 5.2, a feasible N^d has at least one pole. □

5.4 STRUCTURE OF A CU NETWORK

Take two directed paths from x to y, $\pi_R(x,y)$ and $\pi_L(x,y)$, defined in Section 3.5. In Figure 5.1, for example, we get $\pi_R(s^1,t^4)$:$s^1 \to x' \to y' \to s^9 \to s^8 \to t^4$, and $\pi_L(s^4,t^7)$:$s^4 \to w' \to y' \to s^9 \to s^{10} \to t^7$.

If four nodes x,y,w,z are located in the boundary clockwisely in this order and they satisfy $\Pi(x,y)\neq\emptyset$, $\Pi(x,z)\neq\emptyset$, $\Pi(w,y)\neq\emptyset$, $\Pi(w,z)\neq\emptyset$, as illustrated in Figure 5.4, then we call these node are 4-reachable. For example, s^1,t^4,s^4,t^1 in Figure 5.1 are 4-reachable.

Lemma 5.3 In a CU network, if four unilateral nodes x,y,w,z are 4-reachable, then $\pi_R(x,y)$ and $\pi_R(w,z)$ have no node in common. Similarly for $\pi_L(x,z)$ and $\pi_L(w,y)$. Let x' be the node farthest from x among the nodes common to $\pi_R(x,y)$ and $\pi_L(x,z)$. We define y',w',z' similarly. See Figure 5.4. Then the cycle C (neglecting the directions of arcs) consisting of $\pi_R(x',y')$, $\pi_R(w',z')$, $\pi_L(x',z')$ and $\pi_L(w',y')$ is a window in G. Furthermore, V can be divided into two connected divergent sets X^+,W^+ and two connected convergent sets Y^-,Z^- satsfying $x \in X^+$, $y \in Y^-$, $w \in W^+$, $z \in Z^-$.

Proof. Assume that a path $\pi_1 \in \Pi(x,y)$ and a path $\pi_2 \in \Pi(w,z)$ have a common node u. This node u is unique,

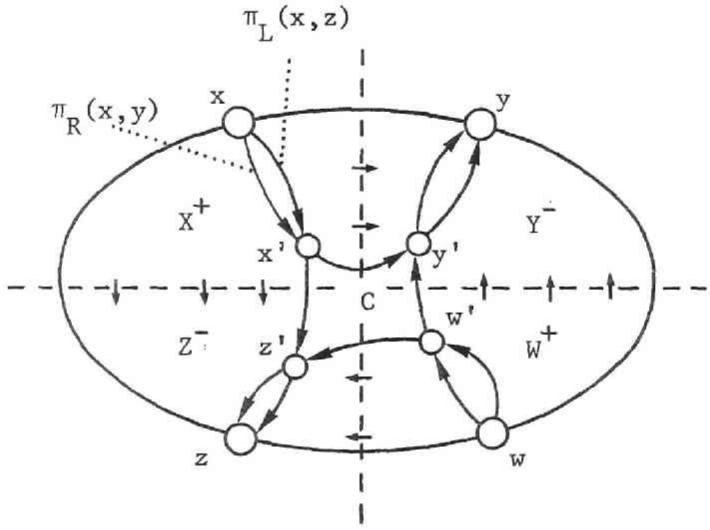


Fig. 5.4 Definition of 4-alternating partition.

since otherwise a directed cycle exists. Node u satisfies $i_u \geq 2$ and $j_u \geq 2$ as shown in Figure 5.4 contradicting the unilaterality of the graph. Therefore $\pi_R(x,y)$ and $\pi_R(w,z)$ have no nodes in common (similarly for $\pi_L(x,z)$ and $\pi_L(w,y)$). The cycle C (neglecting the directions of arcs) consisting of $\pi_R(x',y')$, $\pi_R(w',z')$, $\pi_L(x',z')$ and $\pi_L(w',y')$ is simple by the selection rule of x',y',w' and z'. We show that cycle C is a window in G. If C is not a window, then the area surrounded by C contains an arc $a(u_0,v_0)$. This node v_0 is reachable to a node u_1 in V_B, otherwise the set of nodes reachable from v_0 is a connected convergent set with no nodes in V_B and contradicts Lemma 2.3. Similarly u_0 is reachable from a node u_2 in V_B. The path from u_2 to u_1 through $a(u_0,v_0)$ has nodes u'_1 and u'_2 that belong to C, by planarity of G. Therefore there exists a simple path π_0 from u'_2 to u'_1 through $a(u_0,v_0)$. Without loss of generality, we assume that u'_2 is in $\pi_L(x',z')$ (similarly for the case u'_1 is in the other path except $\pi_L(x',z')$). If u'_1 is in $\pi_L(w',y')$, then the path consisting of $\pi_L(x,u'_2)$, π_0 and $\pi_L(u'_1,y')$ contains an arc located to the right of the first arc in $\pi_R(x',y')$. This contradicts the definition of $\pi_R(x',y')$.

Now add a node v_0 and four arcs $a(x',v_0)$, $a(v_0,y')$,

$a(w',v_0)$, $a(v_0,z')$ in the area surrounded by C. The resulting graph G" still satisfies Assumption A (1)-(3). By Lemma 2.5, $V \cup \{v_0\}$ is divided into two connected divergent sets X^+,W^+, two connected convergent sets Y^-,Z^- and $\{v_0\}$, such that $x \in X^+$, $y \in Y^-$, $w \in W^+$, $z \in Z^-$. These X^+,Y^-,W^+,Z^- also satisfy the condition of the lemma in G. □

We call the above X^+,Y^-,W^+,Z^- a <u>4-alternating partition</u> and call the window C satisfying Lemma 5.3 a <u>partition window</u>. Furhter $A(X^+;Y^-)$ is called a <u>unilateral semi-cut</u>. For example, in Figure 5.1, the partition window C is the cycle containing x',y',w',z' and the 4-alternating partition is given by $X^+=\{s^1,s^7,s^6,s^5,x'\}$, $Y^-=\{t^4,t^9,t^5,s^2,t^2,s^3,t^7,s^8,s^9,s^{10},y'\}$, $W^+=\{s^4,w'\}$ and $Z^-=\{t^1,z'\}$.

A feasible CU network with $S^d \cup T^d \neq \emptyset$ has a pole (x,y) by Lemma 5.2. Since $\Pi(x,y)\neq\emptyset$ implies $\Pi(y,x)=\emptyset$, there exists a unilateral simple cut $A(X^+)$ with $x \in X^+, y \in V-X^+$ by Lemma 2.1. If $r(X^+)=0$ for this connected divergent set X^+, there is no dummy flow from $x \in S^d$ to $y \in T^d$. Similarly, if a 4-alternating partition X^+,Y^-, W^+,Z^- with $x \in X^+, y \in Y^-$ exists for a pole (x,y) and furthermore $r(X^+;Y^-)=0$ holds, then no dummy flow from x to y exists. Here we call a pole

not satisfying these conditions a non-trivial pole. Namely a non-trivial pole (x,y) satisfies the following conditions (i),(ii).

(i) $r(X^+)>0$ for any unilateral simple cut $A(X^+)$ with $x \in X^+, y \in V-X^+$.

(ii) $r(X^+;Y^-)>0$ for any 4-alternating partition X^+, Y^-,W^+,Z^- with $x \in X^+, y \in Y^-$.

For the pole (v_1,v_2) in Figure 5.2, $A(X^+)$ with $X^+=\{v_1,v_6\}$ is a unilateral simple cut, and $v_1 \in X^+, v_2 \in V-X^+, r(X^+)=2-(g^1+g^2)=0$. Therefore (v_1,v_2) is a trivial pole.

Lemma 5.4 If a CU network $N=(G,P,g,c)$ with $S^d \cup T^d \neq \emptyset$ is feasible, then $N^d=(G,P,P^d,g,g^d,c)$ has at least one non-trivial pole.

Proof. See Appendix. □

Here we describe an outline of algorithm MATE2 to be disccussed in Section 5.6. We first construct the set Q of all poles in N^d. If Q contains no non-trivial pole, then N^d is infeasible. Otherwise, there is a positive some amount e of dummy flow from x to y, as will be shown in Section 5.5.

Therefore we create a new commodity K+1 with source sink pair (x,y) and $g^{K+1}=e$. As a result of this, (x,y) is removed from Q. By repeating this until $Q=\emptyset$ holds, we complete the reduction of CU to CB.

5.5 α -CUT AND β-CUT

In this section, we test whether a pole (x_1,y_1) is trivial or not. We also describe some lemmas necessary for determining the amount e(>0) of dummy flow from x to y. Without loss of generality, we assume that no other dummy source or sink exist in the boundary line from x_1 to y_1. When we go around the boundary clockwisely starting from x_1, we define rank(x_1)=0, and rank(z)=i for the i-th node $z\in V_B$. Further let

$$T^d(x_1)\triangleq\{y\in T^d \mid \Pi(x_1,y)\neq\emptyset\}.$$

If $|T^d(x_1)|=1$ (i.e., $T^d(x_1)=\{y_1\}$), then obviously there exists a dummy flow of amount g^d_{x1} from x_1 to y_1. In the subsequent discussion, we consider the case $q=|T^d(x_1)|\geq 2$. Without loss of generality, assume that $T^d(x_1)=\{y_1,y_2,\ldots,y_q\}$ satisfies $\text{rank}(y_1)<\text{rank}(y_2)<\ldots<\text{rank}(y_q)$, as illustrated in Figure 5.5. We further divide V into the following sets.

$$VB_0 \triangleq \{z\in V_B \mid \text{rank}(x_1)(=0)\leq\text{rank}(z)<\text{rank}(y_1)\},$$

$$VB_i \triangleq \{z\in V_B \mid \text{rank}(y_i)\leq\text{rank}(z)<\text{rank}(y_{i+1})\},$$

$$i=1,2,\ldots,q-1,$$

$$VB_q \triangleq \{z\in V_B \mid \text{rank}(y_q)\leq\text{rank}(z)\}.$$

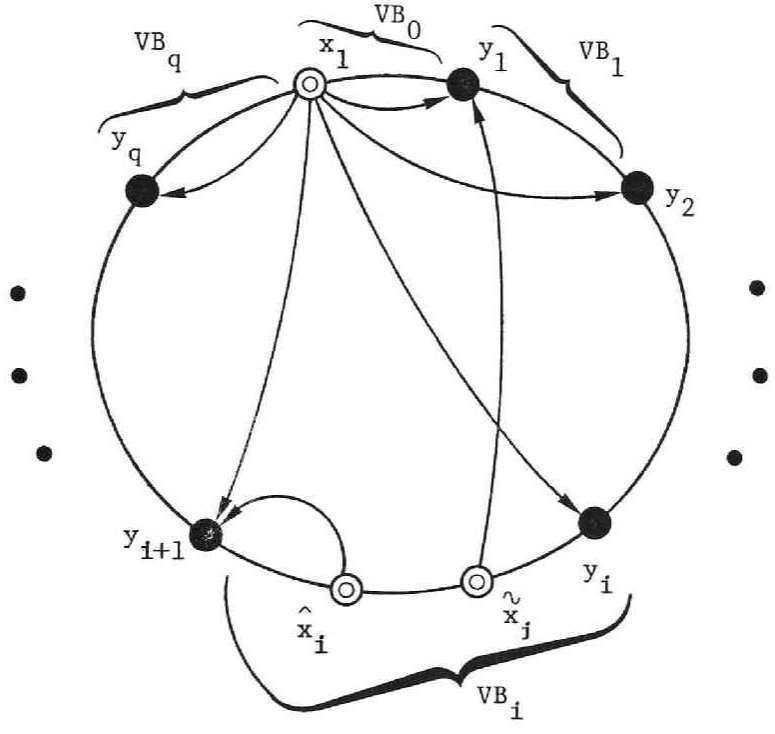


Fig. 5.5 Explanation of Definition of VB_i, $\tilde{x}_i$ and $\hat{x}_i$.

In each VB_i, i=1,2,..,q-1, we define the following sets of dummy sources.

$$\tilde{S}_i^d \triangleq \{x \in S^d | x \in VB_i, \ \Pi(x,y_1)\neq\emptyset\}, \tag{5.2}$$

$$\hat{S}_i^d \triangleq \{x \in S^d | x \in VB_i, \ \Pi(x,y_{i+1})\neq\emptyset\}, \tag{5.3}$$

$$\tilde{I} \triangleq \{i | \tilde{S}_i^d\neq\emptyset, \ i=1,2,\ldots,q-1\}.$$

If $\tilde{S}_i^d\neq\emptyset$, then let $\tilde{x}_i$ denote the node with the maximum rank in $\tilde{S}_i^d\neq\emptyset$. If $\hat{S}_i^d\neq\emptyset$, the let $\hat{x}_i$ denote the node with the minimum rank in $\hat{S}_i^d$. These are also illustrated in Figure 5.5. For example, interpreting pole (s^1,t^4) as (x_1,y_1) in Figure 5.1, we get $y_2=t^5$, $y_3=t^7$, $y_4=t^1$, $y_5=s^4$ (i.e., q=5), and $\tilde{x}_1=t^9$, $\tilde{x}_2=s^2$, $\hat{x}_2=s^3$, $\tilde{x}_3=\hat{x}_3=s^4$, $\tilde{x}_4=s^5$, $\tilde{I}=\{1,2,3,4\}$.

Furthermore, we classify VB_i, i=1,2,..,q-1, as follows.

Case-1: $\tilde{S}_i^d=\emptyset$.

Case-2: $\tilde{S}_i^d\neq\emptyset$ and $\Pi(x_1,\tilde{x}_i)\neq\emptyset$.

Case-3: $\tilde{S}_i^d\neq\emptyset$, $\Pi(x_1,\tilde{x}_i)=\emptyset$ and $\hat{S}_i^d=\emptyset$.

Case-4: $\tilde{S}_i^d\neq\emptyset$, $\Pi(x_1,\tilde{x}_i)=\emptyset$, $\hat{S}_i^d\neq\emptyset$ and $\Pi(x_1,\hat{x}_i)\neq\emptyset$.

There still remains the case of $\tilde{S}_i^d\neq\emptyset$, $\Pi(x_1,\tilde{x}_i)=\emptyset$, $\hat{S}_i^d\neq\emptyset$ and $\Pi(x_1,\hat{x}_i)=\emptyset$. In this case, there exists $X^-(x_1;\tilde{x}_i)$ and $X^-(x_1;\hat{x}_i)$. Let

$$V_i^+ \triangleq (V-X^-(x_1;\tilde{x}_i)) \cup (V-X^-(x_1;\hat{x}_i)).$$

Based on this V_i^+, we add the following cases.

Case-5: $\tilde{S}_i^d \neq \emptyset$, $\Pi(x_1, \tilde{x}_i) = \emptyset$, $\hat{S}_i^d \neq \emptyset$, $\Pi(x_1, \hat{x}_i) = \emptyset$ and $V_i^+ = \emptyset$.

Case-6: $\tilde{S}_i^d \neq \emptyset$, $\Pi(x_1, \tilde{x}_i) = \emptyset$, $\hat{S}_i^d \neq \emptyset$, $\Pi(x_1, \hat{x}_i) = \emptyset$ and $V_i^+ \neq \emptyset$.

Now a unilateral simple cut $A(X_i^+)$ satisfying the following condition is called <u>α-cut</u>.

$y_{i+1}, y_{i+2}, \ldots, y_q, x_1 \in X_i^+$, $y_1, y_2, \ldots, y_i, \tilde{x}_i \in V - X_i^+$, and there are no $x' \in S^d, y' \in T^d$ such that $x' \in VB_i \cap X_i^+$, $y' \in V - X_i^+$ and $\Pi(x', y') \neq \emptyset$.

Here, if $r(X_i^+) = 0$, then any feasible solution f to N^d has no dummy flow from x_1 to y_1 by the definition of r, and therefore (x_1, y_1) is a trivial pole. If $r(X_i^+) > 0$, f has the amount $r(X_i^+)$ of dummy flow through this α-cut, as obvious from the unilateraliy of the cut.

In each of Cases-2,4,5, there always exists an α -cut as shown in the following lemmas.

Lemma 5.5 In Case-2, let $X_i^+ = V - X^-(\tilde{x}_i; y_{i+1})$. Then $A(X_i^+)$ is an α-cut.

Proof. This is illustrated in Figure 5.6. See Appendix for details. □

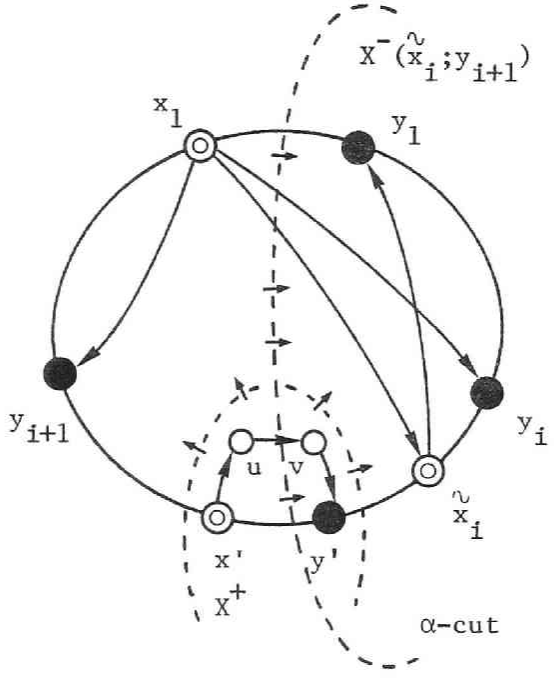


Fig. 5.6 α-cut for Case-2.

Lemma 5.6 In Cases-3,4, let $X_i^+ = X^+(y_{i+1}; \tilde{x}_i)$. Then $A(X_i^+)$ is an α-cut.

Proof. This is illustrated in Figure 5.7. See Appendix for details. □

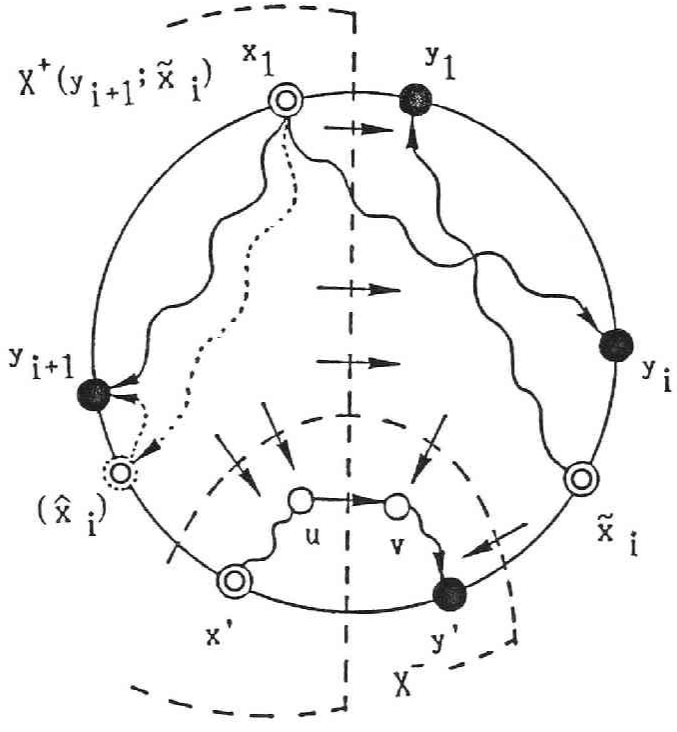


Fig. 5.7 α-cut for Case-3 and 4.

Lemma 5.7 In Case-5, let $X_i^+ = X^+(y_{i+1}; \bar{x}_i) \cup (V - X^-(x_1; \hat{x}_i))$. Then $A(X_i^+)$ is an α-cut.

Proof. This is illustrated in Figure 5.8. See Appendix for details. □

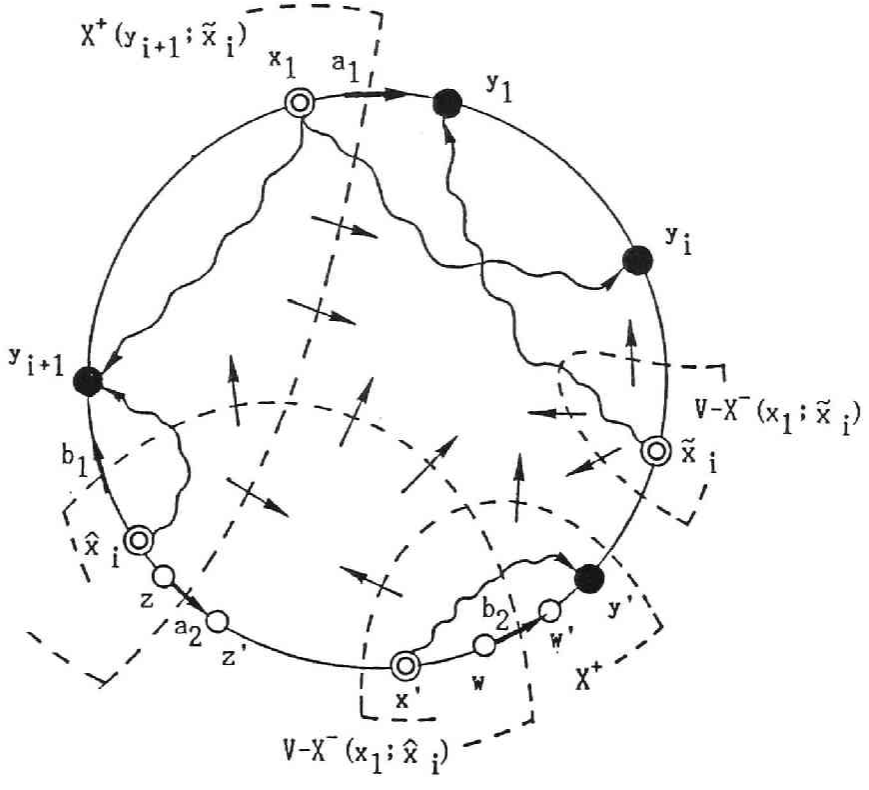


Fig. 5.8 α-cut for Case-5.

For example, let $(x_1,y_1)=(s_1,t_4)$ in Figure 5.1. Then Cases-2,5,3 hold for i=1,2,4, respectively. For i=1,2,4 we obtain

$$X_1^+=V-X^-(\tilde{x}_1;y_2)=V-\{t^4,t^9\},$$

$$X_2^+=X^+(y_3;\tilde{x}_2)\cup(V-X^-(x_1;\hat{x}_2))=\{s^1,x',y',s^9,s^{10},s^3,t^7,$$
$$s^4,t^1,s^5,s^6,s^7,z',w'\}\cup\{s^3\}$$

$$X_4^+=X^+(y_5;\tilde{x}_4)=\{s^1,s^7\}$$

The α-cuts defined for these sets have the following properties.

$$r(X_1^+)=6-(g^4+g^8+g^9)=3,$$

$$r(X_2^+)=7-(g^3+g^4+g^5+g^6+g^9)=2,$$

$$r(X_4^+)=5-(g^1+g^7)=3.$$

In Case-6, there may not be any α-cut. However, there exists a 4-alternating partition X_i^+, Y_i^-, W_i^+, Z_i^+ satisfying the following conditions.

$$x_1\in X_i^+,\qquad y_1,y_2,\ldots,y_i\in Y_i^-,$$
$$\tilde{x}_i,\hat{x}_i\in W_i^+,\quad y_{i+1}\in Z_i^-.$$

We call such a 4-alternating partition of V X_i^+, Y_i^-, W_i^+, Z_i^+ a <u>β-partition</u> for i, and $A(X_i^+;Y_i^-)$ is called a <u>β-semi-cut</u> for i.

Lemma 5.8 Case-6 always has a β-partition for i.

Proof. This is illustrated in Figure 5.10. See Appendix for details. □

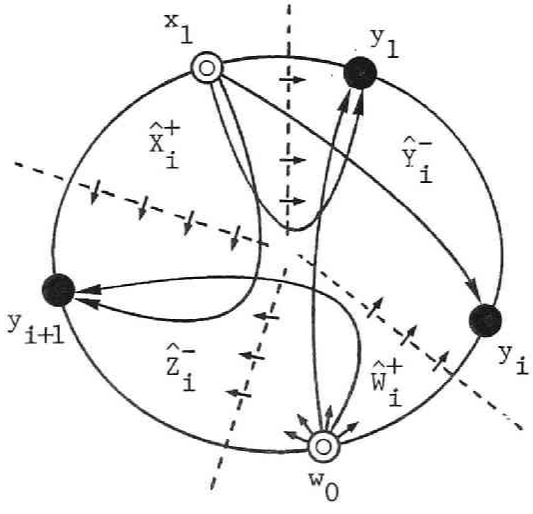


Fig. 5.9 β -partition for Case-6.

Let $(x_1,y_1)=(s^1,t^4)$ in Figure 5.1. Case-6 holds for i=3. A method to construct a β-partition is given in the proofs of Lemmas 5.3 and 5.8. According to them, we obtain

$X_3^+=\{s^1,s^7,s^6,s^5,x'\}$,

$Y_3^-=\{t^4,t^9,t^5,s^2,t^2,s^3,t^7,s^8,s^9,s^{10},y'\}$,

$W_3^+=\{s^4,w'\}$, $Z_3^-=\{t^1,z'\}$.

The resulting β-semi-cut is as follows.

$A(X_3^+;Y_3^-)=\{a(s^1,t^4),a(x',t^4),a(x',s^8),a(x',y')\}$,

$r(X_3^+;Y_3^-)=5-(g^5+g^6+g^7)=2$.

5.6 ALGORITHM MATE2

In this section, we first describe a method for testing the triviality of a pole (x_1,y_1) and for computing the amount $e>0$ of dummy flow from x_1 to y_1 in case (x_1,y_1) is a non-trivial pole. Using this, we then present algorithm MATE2 that determines the assignment of dummy flows.

By the definition, $X^-(\tilde{x}_i;y_{i+1})$, $X^+(y_{i+1};\tilde{x}_i)$ and $X^-(x_1;\hat{x}_i)$ are obtained by computing a maximal directed (or reversely directed) tree with an appropriate root. The corresponding α-cuts can be computed from these. We compute a β-semi-cut defined by a β-partition X_i^+, Y_i^-, W_i^+, Z_i^- as follows. We first obtain $A(X_i^+)$ and compute the β-semi-cut from it as follows. Let x' be the node nearest to y_1 among the nodes contained in both $\pi_R(x_1,y_1)$ and $\pi_L(x_1,y_{i+1})$ (in Figure 5.4, for example, we consider $x=x_1$, $y=y_1$, $z=y_{i+1}$, $x'=x'$). Here $X^+(x';y_1)$ is a cut $A(X_i^+)$ as its 4-reachability can be easily shown. $X^+(x';y_1)$ can be obtained by computing a maximal directed tree. Let C be the partition window providing the β-partition. There exists an arc a with $\{a\}=C\cap A(X_i^+;Y_i^-)$, which can be computed from $A(\pi_R(x_1,y_1))\cap A(X_i^+)$. The β-semi-cut $A(X_i^+;Y_i^-)$ can then be constructed from $A(X_i^+)$ and the arc a.

Lemma 5.9 Let (x_1,y_1) be a pole in a CU network N^d. There are α or β-cuts $A(X_i^+)$ for $i \in \tilde{I}$, such that $X_i^+ \supseteq X_j^+$ holds for every $i<j$. There α, β-cuts for all $i \in \tilde{I}$ of a given pole can be computed in $O(|V|^2)$ time.

Proof. See Appendix. □

In fact, pole $(x_1,y_1)=(s^1,t^4)$ in Figure 5.1 has α-cuts satisfying $X_4^+ \subsetneq X_3^+ \subsetneq X_2^+ \subsetneq X_1^+$.

Let A_i denote the α-cut or β-semi-cut satisfying Lemma 5.9, and let r_i be its r-value.

Lemma 5.10 Take $i.j \in \tilde{I}$ such that $k \notin \tilde{I}$ holds for any k with $i<k<j$. Let e_j denote the amount of dummy flow that goes out of x_1 and passes through cut A_j in a feasible solution. Then the feasible solution has the amount e_i of dummy flow that goes out of x_1 and passes through A_i, where $e_i=\min\{e_j,r_i\}$.

Proof. See Appendix. □

For pole $(x_1,y_1)=(s^1,t^4)$ in Figure 5.1, let $j=4,i=3$. Then we obtain $e_4=r_4=3$, $r_3=2$, $e_3=\min\{e_4,r_3\}=2$ from Lemma 5.10.

Since there exists no α-cut or β-semi-cut for each $i \in \{1,2,\ldots,q-1\}-\tilde{I}$, A_i is undefined for such i. Here, we

assume that the r value of such i is infinite (i.e., let $r_i=+\infty$) for convenience. The amount of any dummy flow from x_1 to y_1 does not clearly exceed the value e determined by

$$e = \min \{g^d_{x1}, r_{q-1}, r_{q-2}, \ldots, r_1, g^d_{y1}\}.$$

The following lemma claims that this amount e of dummy flow from x_1 to y_1 in fact exists.

Lemma 5.11 For a pole (x_1,y_1) in a feasible CU network N^d, assume $q=|T^d(x_1)|\geq 2$, and let

$$e = \min \{g^d_{x1}, r_{q-1}, r_{q-2}, \ldots, r_1, g^d_{y1}\}.$$

Then a feasible solution in N^d has the amount e of dummy flow from x_1 to y_1.

Proof. Apply Lemma 5.10 in the order of i=q-1, q-2,...,1. See Appendix for details. □

For pole $(x_1,y_1)=(s^1,t^4)$ in Figure 5.1, we obtain e=4 from $g^d_{x1}=4$, $g^d_{y1}=4$, $r_4=3$, $r_3=2$, $r_2=2$, $r_1=3$.

Lemma 5.12 A pole (x_1,y_1) in a feasible CU network N^d is non-trivial if and only if e>0.

Proof. If e=0, it is clearly trivial. If e>0, there exists a positive dummy flow from x_1 to y_1 by Lemma 5.11. Therefore conditions (i),(ii) for being a non-trivial pole (described prior to Lemma 5.4) are satisfied. □

In the following algorithm for reducing a CU network N^d to a CB network, we compute the e of Lemma 5.11 for each pole (x_1,y_1). If $e<0$ holds for some pole (x_1,y_1), then N^d is infeasible and we terminate the procedure. If $e=0$, then (x_1,y_1) is a trivial pole and we remove it from the set of poles Q. If $e>0$, then the amount e of dummy flow from x_1 to y_1 is replaced by a new commodity K+1 applying the following procedure NEWITEM.

Procedure NEWITEM(x_1,y_1,e,K)

$s^{K+1}:=x_1$, $t^{K+1}:=y_1$, $g^{K+1}:=e$,

$S:=S\cup\{s^{K+1}\}$, $T:=T\cup\{t^{K+1}\}$, $P:=P\cup\{(s^{K+1},t^{K+1})\}$,

$g^d_{x1}:=g^d_{x1}-e$, $g^d_{y1}:=g^d_{y1}-e$,

$S^d:=S^d-\{s^d_{x1}\}$. if $g^d_{x1}=0$.

$T^d:=T^d-\{t^d_{y1}\}$, if $g^d_{y1}=0$. □

For pole $(x_1,y_1)=(s^1,t^4)$ in Figure 5.1, the 11-th commodity with $(s^{11},t^{11})=(s^1,t^4)$, $g^{11}=2$ is created by NEWITEM($s^1,t^4,2,10$).

As a result of this, if there exists an i with $e=r_i$, then the updated r_i becomes 0. Therefore pole (x_1,y_1) is removed from Q. If $e=g^d_{x1}$ (or $e=g^d_{y1}$), then $x_1 \notin S^d$ ($y_1 \notin T^d$) implies that (x_1,y_1) is not a pole any longer, and it is

removed from Q. At this time, it is possible that a new neighbouring pair (x',y_1) (or $(x_1,y'),(x',y')$) is generated; we add such a pole to Q. From the above discussion, we obtain the following algorithm MATE2 in order to reduce CU networks to CB networks.

Procedure MATE2

Input: CU network $N=(G,P,g,c)$, where $g^k=(g^1,g^2,\ldots,g^K)$.

Output: CB network $\tilde{N}=(G,\tilde{P},\tilde{g},c)$ that is feasible if and only if so is N, where $\tilde{g}^k=(g^1,g^2,\ldots,g^K,\ldots,g^{\tilde{K}})$.

0. Construct $N^d=(G,P,P^d,g,g^d,c)$ with commodities $k=1,2,\ldots,K,d.$ and compute the reachiability $\Pi(x,y)$ between every pair of $x\in S^d$ and $y\in T^d$. The set of all such $\Pi(x,y)$ is denoted $\Pi(S^d;T^d)$. From $\Pi(S^d;T^d)$, compute the set of poles Q in N^d.

1. If $Q=\emptyset$ and $S^d\cup T^d\neq\emptyset$, halt by concluding infeasibility. If $Q=\emptyset$ and $S^d\cup T^d=\emptyset$ (CB network is obtained), let $\tilde{P}:=P$, $\tilde{K}:=K$ and halt.

2. IF $Q\neq\emptyset$, then choose a pole (x_1,y_1) in Q and compute $T^d(x_1)$, $q=|T^d(x_1)|$. If q=1, let $e:=\min\{g^d_{x1},g^d_{y1}\}$ and proceed to 3. If $q\geq 2$, compute $\tilde{S}^d_i$, $\hat{S}^d_i$ $(i=1,2,\ldots,q-1)$ from $\Pi(S^d;T^d)$. Then compute α-cuts or β-semi-cuts A_i for $i\in\tilde{I}$, as well as thier values r_i. Let

$e:=\min\{g^d_{x1}, r_{q-1}, r_{q-2}, \ldots, r_1, g^d_{y1}\}$.

If e=0, proceed to 4. If e<0, then halt by concluding infeasibility.

3. Create a new commodity by calling NEWITEM(x_1, y_1, e, K).

4. $Q:=Q-\{(x_1,y_1)\}$. If a new neighbouring pair (x',y_1) (or (x_1,y'),(x',y')) is generated, check whether the neighbouring pair is a pole or not by using $\Pi(S^d;T^d)$. Add it to Q if it is a pole. Return to 1. □

The initial cardinality of Q is at most $\min\{|S^d|,|T^d|\}*2$ because each dummy sorce (sink) is in V_B and contained in at most two poles. This cadinality decreases by one at Step 4, or does not change when a new pole is created in Step 4. If $|Q|$ does not change, however, the cardinality of $S^d \cup T^d$ decreases at least by one. This means that the number of iterations until $Q=\emptyset$ is at most the sum of the initial cardinality of Q and $|S^d \cup T^d|$, which is $O(|V_B|)$. Since the amount e of Lemma 5.11 can be determined in $O(|V|^2)$ time for a pole $(x_1,y_1) \in Q$ by Lemma 5.9, the time required in one cycle of Steps 1-4 is also $O(|V|^2)$. $\Pi(S^d;T^d)$ in Step 0 can be computed in $O(|V|^2)$ time. Therefore the entire time of MATE2 is bounded from above by $O(|V|^3)$. The required space is $O(|V|^2)$ which

is necessary to store $\Pi(S^d;T^d)$ and other data. Fainally we consider the number of new commodities created by MATE2. Since the source sink pair of any new commodity is a neighbouring pair when it is created by MATE2, no two paths from sources to sinks of new commodities cross each other. Thus the number of the created commodities is bounded from above by $|V_B|$. This proves the following theorem.

Theorem 5.1 If a CU network N=(G,P,g,c) of Definition 5.1 is feasible, then a feasible solution to N satisfies the integral flow property. The procedure MATE2 and ASSIGN test the feasibility of a CU network correctly. The required time and space are $O(K|V|+|V|^3)$, $O(K|V|+|V|^2)$ respectively. □

For the network N_A in Figure 5.1, we obtain the initial set of poles $Q=\{(s^1,t^4),(t^9,t^4),(s^2,t^5),(s^3,t^7),(s^4,t^7),(s^4,t^1),(s^5,t^1),(s^1,s^7)\}$. We illustrate the process of MATE2 applied to N_A in Table 5.1. New commodities k=11-20 are created.

Table 5.1 Application of MATE2 to N_A.

Selected pole (x_1, y_1)	$g_{x1}{}^d, g_{y1}{}^d$	q	Case and r_i for each i	e	New item	Out from S^d	Out from T^d	Trivial pole after modification	Pole added to Q
(s^1, t^4)	4, 4	5	i=1:Case-2, r_1=3		K=11			(s^1, t^4)	
			i=2:Case-5, r_2=2						
			i=3:Case-6, r_3=2						
			i=4:Case-3, r_4=3	2					
(t^9, t^4)	1, 2	1		1	K=12	t^9			
(s^2, t^5)	2, 1	2	i=1:Case-1, r_1=+∞	1	K=13		t^5		(s^2, t^4)
(s^2, t^4)	1, 1	1		1	K=14	s^2	t^4		
(s^3, t^7)	2, 4	1		2	K=15	s^3			(s^1, t^7)
(s^1, t^7)	2, 2	2	i=1:Case-6, r_1=0 ※1	0				(s^1, t^7)	
(s^4, t^7)	4, 2	2	i=1:Case-6, r_1=2 ※2	2	K=16		t^7		
(s^4, t^1)	2, 5	1		2	K=17	s^4			(s^1, t^1)
(s^1, t^1)	2, 3	2	i=1:Case-3, r_1=1 ※3	1	K=18			(s^1, t^1)	
(s^5, t^1)	2, 2	1		2	K=19	s^5	t^1		
(s^1, s^7)	1, 1	1		1	K=20	s^1	s^7		

※1: $A(X_1{}^+; Y_1{}^-) = \{a(s^1, t^4), a(x', t^4), a(x', s^8), a(x', y')\}$

※2: $A(X_1{}^+; Y_1{}^-) = \{a(w', y'), a(w', s^{10}), a(s^4, t^7)\}$

※3: $A(X_1{}^+) = \{a(s^1, t^4), a(s^1, x'), a(s^7, x'), a(s^7, s^6)\}$

5.7 ON THE CONDITIONS DEFINING CU NETWORKS

If we remove the conditions (3) and (4) of Definition 5.1, the network in Figure 5.10 serves as a counterexample to the integral flow property. This suggests the difficulty to generalize the method discussed in this chapter to such networks.

For any positive rational number k/n (n and k are positive integers such that $1 \leq k \leq n$), we can construct a network that has the unique feasible flow with its flow value of a commodity being equal to k/n, as shown in Figure 5.10. This network has $2n^2+4$ nodes, $3n^2+2n$ arcs and 2 commodities, and all capacities of arcs are 1 and $g^1=k$ and $g^2=n-k$. Furthermore, the network is planar and acyclic, and all sources and sinks are located in the boundary, that is, Assumption A is satisfied. The feasible solution f of the network is given as follows, where only non-zero flows are described.

$f(a(s^1,u_{i1}),1)=k/n$ for $i=1,2,\ldots,n$,

$f(a(v_{in},t^1),1)=k/n$ for $i=1,2,\ldots,n$,

$f(a(u_{ij},v_{ij}),1)=k/n$ for $i=1,2,\ldots,n$, $j=1,2,\ldots,n$,

$f(a(v_{ij},u_{ij+1}),1)=k/n$ for $i=1,2,\ldots,n$, $j=1,2,\ldots,n-1$,

$f(a(s^2,u_{1i}),2)=(n-k)/n$ for $i=1,2,\ldots,n$,

$f(a(v_{ni},t^2),2)=(n-k)/n$ for $i=1,2,\ldots,n$,

$f(a(u_{ij},v_{ij}),2)=(n-k)/n$ for i=1,2,...,n, j=1,2,...,n,

$f(a(v_{ij},u_{i+1j}),2)=(n-k)/n$ for i=1,2,....,n-1.

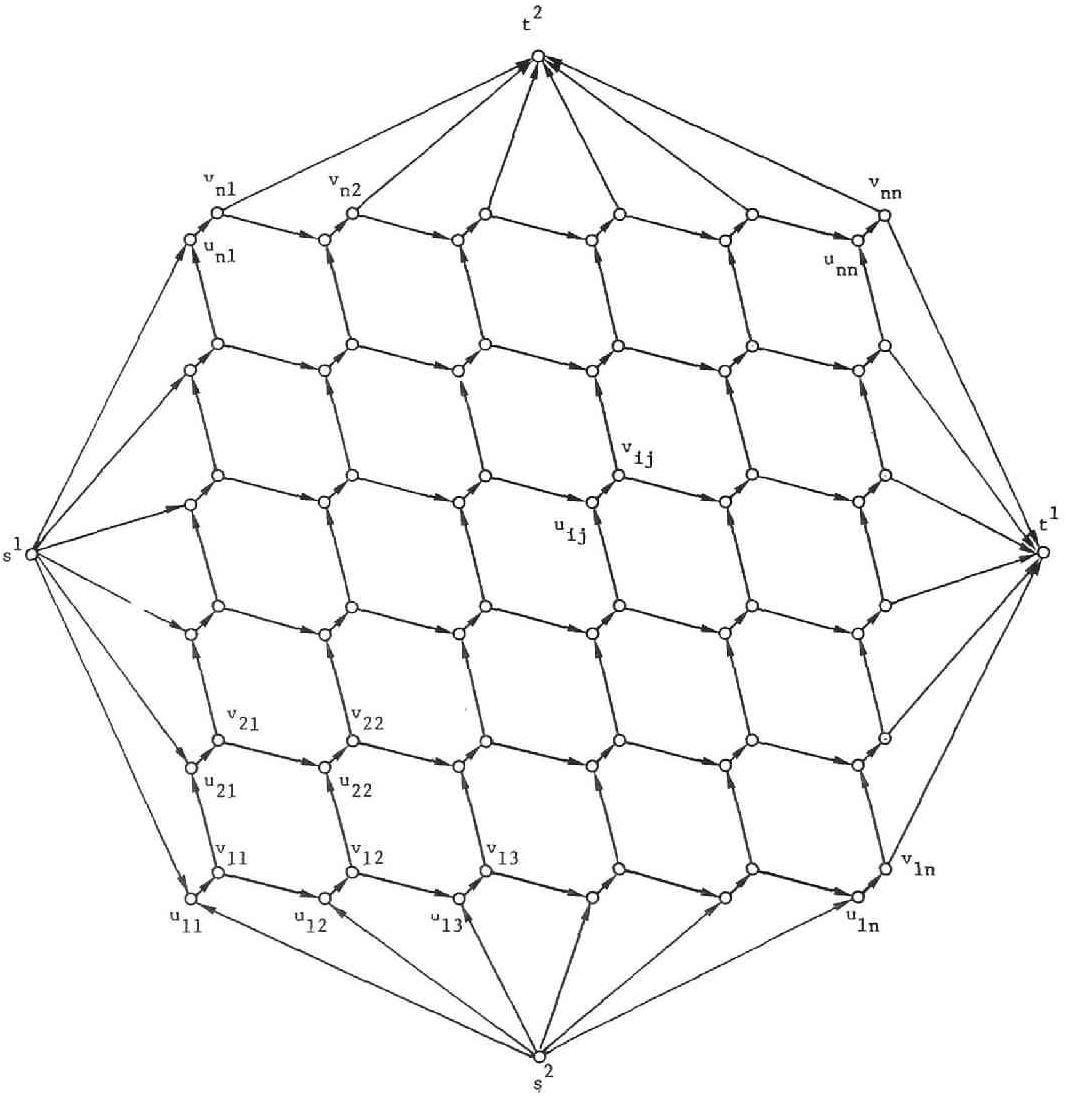


Fig. 5.10 A counterexample to integral flow property.

The uniqueness of the feasible flow is proved as follows. Assume the existence of a feasible flow f' through paths $\pi_1, \pi_2, \ldots, \pi_p \in \Pi(s^1,t^1)$, and $\pi'_1, \pi'_2, \ldots, \pi'_q \in \Pi(s^2,t^2)$ and nonnegative real values $e_1, e_2, \ldots, e_p$, and $e'_1, e'_2, \ldots, e'_q$ such that

$$f'(a,1) = \Sigma\{e_i \mid A(\pi_i) \text{ contains arc } a\},$$

$$f'(a,2) = \Sigma\{e'_i \mid A(\pi'_i) \text{ contains arc } a\},$$

$$\Sigma\{e_i \mid i=1,2,\ldots,p\} = k,$$

$$\Sigma\{e'_i \mid i=1,2,\ldots,q\} = n-k.$$

Here, let $A_k^1 = \{a(u_{ik}, v_{ik}) \mid i=1,2,\ldots,n\}$ and $A_k^2 = \{a(u_{ki}, v_{ki}) \mid i=1,2,\ldots,n\}$ for $k=1,2,\ldots,n$. For any k, if we remove the set of arcs A_k^1 from the network, then source s^1 becomes unreachable to sink t^1. Therefore, any path π in $\Pi(s^1,t^1)$ satisfies $A(\pi) \cap A_k^1 \neq \emptyset$ for $k=1,2,\ldots,n$, and hence $|A(\pi_i) \cap A_k^1| \geq 1$ holds for $i=1,2,\ldots,p$, $k=1,2,\ldots,n$. Similarly we have $|A(\pi'_i) \cap A_k^2| \geq 1$ for $i=1,2,\ldots,q$, $k=1,2,\ldots,n$. Since the sum of flows of commodity 1 in A_k^1 cannot exceed the sum of capacities in A_k^1, properties $|A(\pi_i) \cap A_k^1| \geq 1$, $i=1,2,\ldots,p$, $k=1,2,\ldots,n$, gives the following,

$$n \geq \sum_{i=1}^{p} |A(\pi_i) \cap A_k^1| e_i \geq \sum_{i=1}^{p} e_i = k \quad \text{for } k=1,2,\ldots,n.$$

Similarly we obtain

$$n \geq \sum_{i=1}^{q} |A(\pi'_i) \cap A_k^2| e'_i \geq \sum_{i=1}^{q} e'_i = n-k \quad \text{for } k=1,2,\ldots,n.$$

Since the sum of flows of commodities 1 and 2 in $\{a(u_{ij},v_{ij}) | i=1,2,\ldots,n, j=1,2,\ldots,n\}$ $(= \bigcup_{k=1}^{p} A_k^1 \cup \bigcup_{k=1}^{q} A_k^2)$ does not exceed the sum of capacities of $\{a(u_{ij},v_{ij}) | i=1,2,\ldots,n,\ j=1,2,\ldots,n\}$, we have

$$n^2 \geq \sum_{k=1}^{n} \sum_{i=1}^{p} |A(\pi_i) \cap A_k^1| e_i + \sum_{k=1}^{n} \sum_{i=1}^{q} |A(\pi'_i) \cap A_k^2| e'_i$$

$$\geq n \sum_{i=1}^{p} e_i + n \sum_{i=1}^{q} e'_i = nk + n(n-k) = n^2.$$

This implies

$$|A(\pi_i) \cap A_k^1| = 1 \text{ for } i=1,2,\ldots,p,\ k=1,2,\ldots,n,$$

$$|A(\pi'_i) \cap A_k^2| = 1 \text{ for } i=1,2,\ldots,q,\ k=1,2,\ldots,n.$$

Without loss of generality, we can assume p=q=n. In this case, π_i,i=1,2,..,n can be written as

$$s^1 \to u_{i1} \to v_{i1} \to u_{i2} \to v_{i2} \to \ldots \to u_{in} \to v_{in} \to t^1, \quad i=1,2,\ldots,n,$$

and π'_i,i=1,2,..,n can be written as

$$s^2 \to u_{1i} \to v_{1i} \to u_{2i} \to v_{2i} \to \ldots \to u_{ni} \to v_{ni} \to t^2, \quad i=1,2,\ldots,n.$$

Here, since the flow of commodities 1 and 2 in each arc $a(u_{1i},v_{1i}) \in A_1^2$ cannot exceed its capacity 1, we have

$$1 \geq e_1 + e'_i \quad \text{for } i=1,2,\ldots,n,$$

which then implies

$$n \geq ne_1 + \sum_{i=1}^{n} e'_i = ne_1 + n-k.$$

This proves $e_1 \leq k/n$. Similarly we get $e_i \leq k/n$, $i=1,2,\ldots,n$. That is, by $\Sigma\{e_i \mid i=1,2,\ldots,p\}=k$, we obtain

$$f(a(u_{ij},v_{ij}),1)=k/n \quad \text{for } i=1,2,\ldots,n,\ j=1,2,\ldots,n.$$

The uniqueness of the values of other flows can be shown similarly. This proves that the solution is unique.

Lastly we show that the max-flow min-cut theorem does not hold for CU networks. The counterexample to the max-flow min-cut theorem presented by Kennington [KENN 78] is shown in Figure 5.2. This network N_B happens to be a CU network as easily verified. Although this network satisfies the cut condition, it is infeasible. In fact, MATE2 and ASSIGN reveal its infeasibiliy as follows. MATE2 decomposes dummy flows from S^d to T^d in N_B into new commodities k=3-5 such that $g^3=g^4=g^5=1$, $(s^3,t^3)=(v_1,v_6)$, $(s^4,t^4)=(v_5,v_2)$, $(s^5,t^5)=(v_3,v_4)$. When ASSIGN is applied to the resulting CB network $\tilde{N}_B$, ASSIGN halts at node v_5 by concluding infeasibility. This infeasibility may be explained as follows. The minimum cut to $\tilde{N}_B$ is $C=\{a_1,a_2,a_3\}$. By removing C, node set V is divided into $X=\{v_1,v_2,v_3\}$ and $V-X=\{v_4,v_5,v_6\}$ and

$$\tilde{r}(X)=c(X)-\sum g^k=2-(g^2+g^3+g^5)=-1<0$$

holds. Thus $\tilde{N}_B$ does not satisfy the cut condition. Among the commodities 2,3 and 5 that indicate infeasibility here, commodities k=3,5 are created from dummy flows by MATE2. This cut satisfies the cut-condition (2.1) in the original network N_B,

$$r(X)=\tilde{r}(X)+g^3+g^5=-1+2=1>0.$$

In other words, testing the cut condition can not find the infeasibilty before specifying a one-to-one correspondence between all dummy sources and sinks.

5.8 CONCLUSION

Among the known classes of multicommodity flow problems in directed networks for which it is possible to develop efficient graph theoretic algorithms, the class of CU networks possesses a unique characteristic that it does not satisfy the max-flow min-cut theorem. It should also be noted that class CU does not contain class CS, because CU must satisfy the unilaterality property which is not imposed on CS. In the next section, we show the max-flow min-cut theorem holds for classes CB and CS.

CHAPTER 6

MAX-FLOW MIN-CUT THEOREM FOR CLASSES CB AND CS

6.1 INTRODUCTION

In this chapter, we first show that the max-flow min-cut theorem holds for class CB. For this purpose, an $O(K|V|)$ time algorithm is constructed to find a minimum cut not satisfying the cut condition for an infeasible CB network. This result is then extended to class CS. Class CU, however, does not have this property [NAGA 88a].

6.2 MAX-FLOW MIN-CUT PROPERTY

Let VB[x,y] denote the set of nodes in B(x,y) (where we assume $x,y \in$ VB[x,y]). If nodes $v,z \in V_B$ appear along B(x,y)

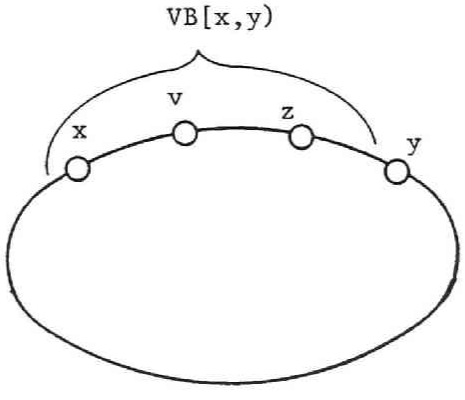


Fig. 6.1 Definition of left-right relation among the nodes in VB[x,y).

in the order of x,v,z,y, then v is to the left of z in B(x,y) (see Figure 6.1). Further we introduce notations

$$VB(x,y]=VB[x,y]-\{x\},\ VB[x,y)=VB[x,y]-\{y\},\ \text{and}$$

$$VB(x,y)=VB[x,y]-\{x,y\}.$$

It is clear that the cut-condition (2.1) for the multicommodity flow problem is nescessary for a network to be feasible, as described in Section 2.4. In general, however, the cut-condition alone is not sufficient to guarantee the feasibility of the multicommodity flow problem. For example, the networks in Figure 6.2 and 6.3 satisfy the cut-condition, but they are infeasible. Note that the networks in Figure 6.2 and 6.3 satisfy Assumption C (i.e., $\Delta\, c(x)=0$ for all $x \in V$) and have the unique source and sink for each commodity. Furthermore, the network in Figure 6.2 is acyclic.

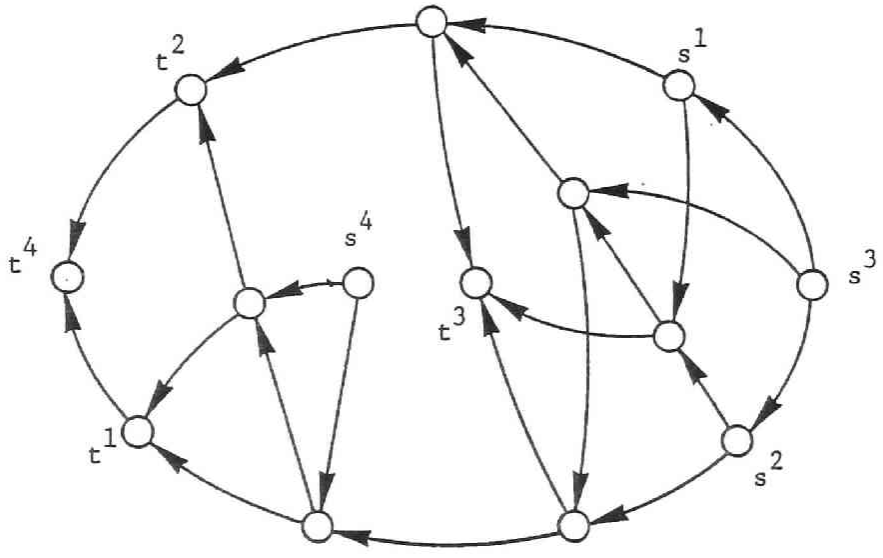


Fig. 6.2 A counterexample to max-flow min-cut property.

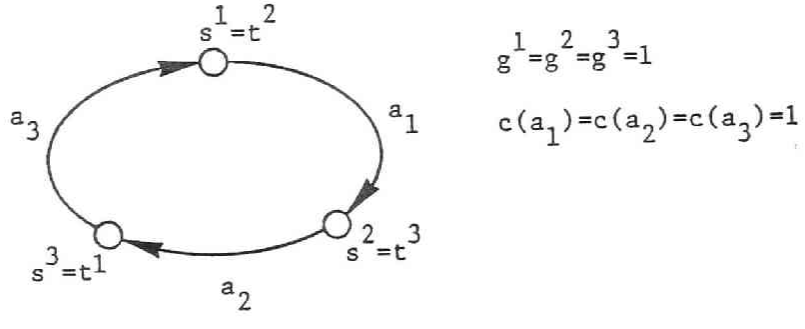


Fig. 6.3 A counterexample to max-flow min-cut property.

If the cut-condition is a sufficient condition that a network in some class is feasible, we call the max-flow min-cut theorem holds for the class (or the class has the max-flow min-cut property). In this chapter, we show the following theorems.

Theorem 6.1 The max-flow min-cut theorem holds for class CB. □

Theorem 6.2 The max-flow min-cut theorem holds for class CS. □

6.3 PROPERTIES OF AN INFEASIBLE NETWORK

Given an infeasible network $N''=(G'',P'',g'',c'')$, assume that ASSIGN halts at node x^* by (T-4) in Step 3 since $a(x^*,v^*)\in OUT_p(x^*)$, $f(a(x^*,v^*),k^*)>0$, $\Pi(v^*,t^{k*})=\emptyset$. Based on this information, we now clarify some properties of N'' in this section. These will be used in the subsequent discussion to show that there is a cut which does not satisfy the cut-condition. A concrete method to construct such a cut will be developed in the next section.

Let f denote the flow assigned in N'' at the time when ASSIGN halts as described above. Clearly f is assigned to the arcs in $OUT(SCAN(x^*))\cup OUT(x^*)$. (The flow in $OUT(x^*)$ is also available because ASSIGN has already passed through Step 2 for x^*.) Now we define $w,z,w',z',w'',z'',Z^+,Z^-$ for x^* in a manner similar to the proof of Lemma 3.5 as shown in Figure 3.2 (we consider $X_q^+=X_{q+1}^+$ for $IN(x^*)=IN_1(x^*)$).

For simplicity, we transform the CB network $N''=(G'',P'',g'',c'')$ into the following network.

Definition 6.1 $REACH(x^*)$ is the set of nodes in a maximal directed tree with root x^* in $G''=(V'',A'')$. Obviously $REACH(x^*)$ is a connected divergent set. The set $V''-REACH(x^*)$ consists of connected convergent sets $\hat{X}_i^-$, $i=1,2,\ldots,q$, where

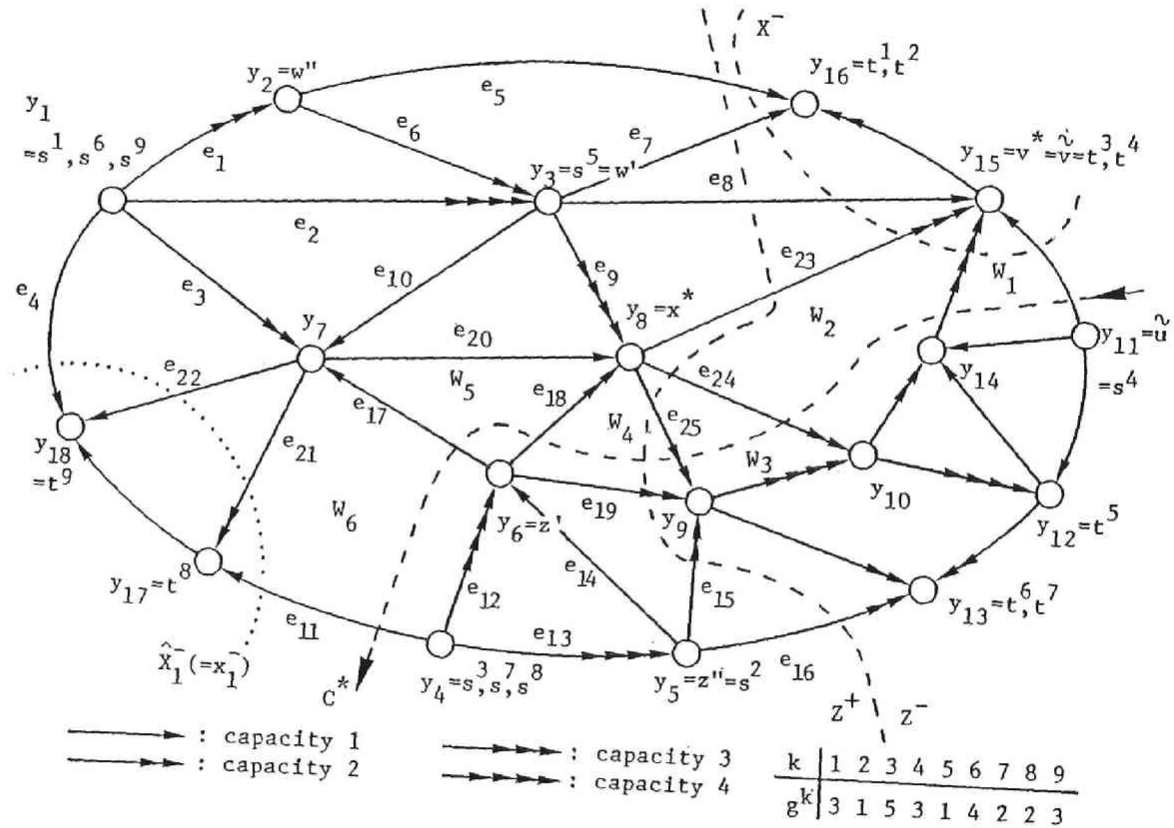


Fig. 6.4 CB network N_1.

Table 6.1 Flows f obtained by ASSIGN for N_1

k	1	2	3	4	5	6	7	8	9
e_1	3								
e_2						4			
e_3									2
e_4									1
e_5	1								
e_6	2								
e_7	1								
e_8	1								
e_9				1		3			
e_{10}						1			
e_{11}								1	
e_{12}			3					1	
e_{13}			2				2		

k	1	2	3	4	5	6	7	8	9
e_{14}		1							
e_{15}			2						
e_{16}							2		
e_{17}								1	
e_{18}		1	1						
e_{19}			2						
e_{20}						1			
e_{21}								1	1
e_{22}									1
e_{23}		1	1		1				
e_{24}						2			
e_{25}						2			

some $\hat{X}_i^-$ is equal to Z^- (we assume $\hat{X}_q^-=Z^-$ without loss of generality). Let G=(V,A) be the resulting graph obtained by shrinking each set $\hat{X}_i^-$,i=1,2,...,q-1, to node x_i^-, except Z^-. That is, $Z^+=\{x_1^-,x_2^-,\ldots,x_{q-1}^-\} \cup \mathrm{REACH}(x^*)$ holds in G. Let P=P" and g=g" and c(a)=c"(a) for $a \in A$. We put all sources and sinks contained in $\hat{X}_i^-$ on x_i^-, i.e., for i=1,2,...,q-1. Let $s^k=x_i^-$ ($t^k=x_i^-$) in N if $s^k \in \hat{X}_i^-$ ($t^k \in \hat{X}_i^-$) in N". In particular, let P=P-(s^k,t^k) and $g^k=0$ for each k with $s^k=t^k=x_i^-$. Let N=(G,P,g,c) denote the resulting network. Then N is also a CB network by Lemma 2.6. We may consider the case in which flow in OUT(x") has already been assigned for some x" before ASSIGN scans x^*, where x" is a node in Z^-. Here, obviously x" is not reachable to x^* and the level of x" should be lower than that of x^*. In this case, no flow in Z^- is necessary to determine the flow in OUT(REACH(x^*)). For simplicity, therefore, we assume that all node in Z^- are unscanned. That is, in network N, flow in an arc a is not determined (f(a,k)=0) if and only if the initial node of a is in $Z^-(=\hat{X}_q^-)$. Let (N,f) be the pair of the resulting network N and the flow f in N. We define

$$K(a)=\{k \mid f(a,k)>0\},\ a \in A,$$

$$TK(a)=\{t^k \mid f(a,k)>0\},\ a \in A. \ \square$$

For example, apply ASSIGN to the network in Figure 6.4. The arcs in each $OUT(y_i)$,i=1,4,11,... are assigned flows in the order of $e_1,e_2,\ldots,e_{25}$. When ASSIGN chooses node y_8, it halts by concluding infeasibility because $f(a(y_8,y_{15}),5)>0$ and $\Pi(y_{15},t^5)=\emptyset$ hold in Step 3. Table 6.1 shows the flows assigned before ASSIGN chooses node y_8. According to the above definition, we have

$REACH(x^*)=\{y_1,y_2,y_3,y_4,y_5,y_6,y_7,y_8\}$,

$OUT_p(x^*)=\{e_{23},e_{24},e_{25}\}$,

$x^*=y_8$, $v^*=y_{15}$, $k^*=5$, $w''=y_2$, $w'=y_3$, $z''=y_5$, $z'=y_6$,

$Z^+=\{\{y_1,y_2,y_3,y_4,y_5,y_6,y_7,y_8,y_{17},y_{18}\}$,

$Z^-=\{y_9,y_{10},y_{11},y_{12},y_{13},y_{14},y_{15},y_{16}\}$,

$V''\text{-}REACH(x^*)=\hat{X}_1^- \cup \hat{X}_2^-$,

$\hat{X}_1^-=\{y_{17},y_{18}\}$ ($\hat{X}_1^-$ will be shrunk into node x_1^-),

$\hat{X}_2^-=Z^-$.

Lemma 6.1 In (N,f), $TK(a(u,v))=\emptyset$ (i.e., $K(a(u,v))=\emptyset$) holds if and only if $u \notin REACH(x^*)$.

Proof. Obvious from $c(a)>0$, $a \in A$ and Definition 6.1. □

By applying Lemma 2.6 to i=1,2,...,q-1 repeatedly, we see that Assumption A also holds in G. Furthermore, divergent nodes x_i^-, i=1,2,...,q-1, are located on the

boundary of G.

Since ASSIGN halts by (T-4) at x^* in N", we have v^*,k^* and p for which $a(x^*,v^*)\in OUT_p(x^*)$, $k^*\in K(a(x^*,v^*))$ and $\Pi(v^*,t^{k*})=\emptyset$ hold in N. See Figure 6.5. Let X be the set of nodes reachable from v^*. By applying the same argument in the proof of Lemma 2.1 to $\Pi(v^*,t^{k*})=\emptyset$, we have a connected divergent set X^+ such that $t^{k*}\in X^+\subseteq V-X$ holds and set $X^-=V-X^+$ is connected. X^- contains v_0, but not x^*. Therefore $X^-\subseteq Z^-$.

In (N,f), a cut $A(X)\cup A(V-X)$ dividing V into X and V-X is called <u>commodity disjoint</u> if the following conditions hold.

$A(X)\subseteq OUT(REACH(x^*))$ (i.e., flow in A(X) has already determined.) (6.1)

$K(a)\cap K(b)=\emptyset$ for $a\in A(X)$ and $b\in A(V-X)$. (6.2)

$TK(a)\subseteq V-X$ for $a\in A(X)$. (6.3)

Let $A(X)\cup A(V-X)$ be a commodity disjoint cut. Since all arcs in A(X) are saturated, we have,

$$c(X)=\sum_{k\in K(a)}\ \sum_{a\in A(X)} f(a,k).$$

From (6.3),

$$g(X)\geq \sum_{k\in K(a)}\ \sum_{a\in A(X)} f(a,k)$$

holds, and hence

$$r(X)=c(X)-g(X)=\sum_{k\in K(a)}\sum_{a\in A} f(a,k)-g(X)\leq 0.$$

Therefore unless $r(X)=0$, the cut-condition (2.1) is not satisfied. Algorithm FIND-CUT described in the subsequent discussion is a procedure to find out such a commodity disjoint cut not satisfying the cut-condition.

Definition 6.2 All devergent nodes reachable to s^{k*} are contained in $VB[z'',w''](=Z^{+}\cap V_B)$. Let $DIV(Z^{+})$ denote the set of divergent nodes in Z^{+}. Let the unilateral simple cut $A(V-X^{-})$ be denoted by $\{b_1,b_2,\ldots,b_m\}$ as shown in Figure 6.5, where $A(V-X^{-})\cap A_B=\{b_1,b_m\}$ and the direction of arc $b_1=a(\tilde{u},\tilde{v})$ is counterclockwise along the boundary. □

For example, let $x^{*}=y_8, v^{*}=y_{15}$ in Figure 6.4. Then we have $X^{-}=\{y_{15},y_{16}\}$, $b_1=a(y_{11},y_{15})$, $b_2=a(y_{14},y_{15})$, $b_3=e_{23}$, $b_4=e_8$, $b_5=e_7$, $b_6(=b_m)=e_5$, $\tilde{u}=y_{11}$, $\tilde{v}=y_{15}$.

Lemma 6.2 For a given (N,f), we define k^{*}, Z^{-} and X^{-} as described in the above. Then $t^{k*}\in Z^{-}-X^{-}$ holds, as shown in Figure 6.5.

Proof. $t^{k*}\in V-X^{-}$ is obvious from $\Pi(v^{*},t^{k*})=\emptyset$. Since ASSIGN has passed Step 2 when x^{*} is chosen, $k^{*}\in H_p\subseteq K_p(x^{*})$ holds. Note that $K_p(x^{*})$ is the set of commodities k whose

sinks $t^k(\neq x^*)$ are reachable from x^* via $OUT_p(x^*)$. Therefore such sinks t^k are contained in Z^- by divergency of Z^+. This implies $t^{k*} \in Z^-$. □

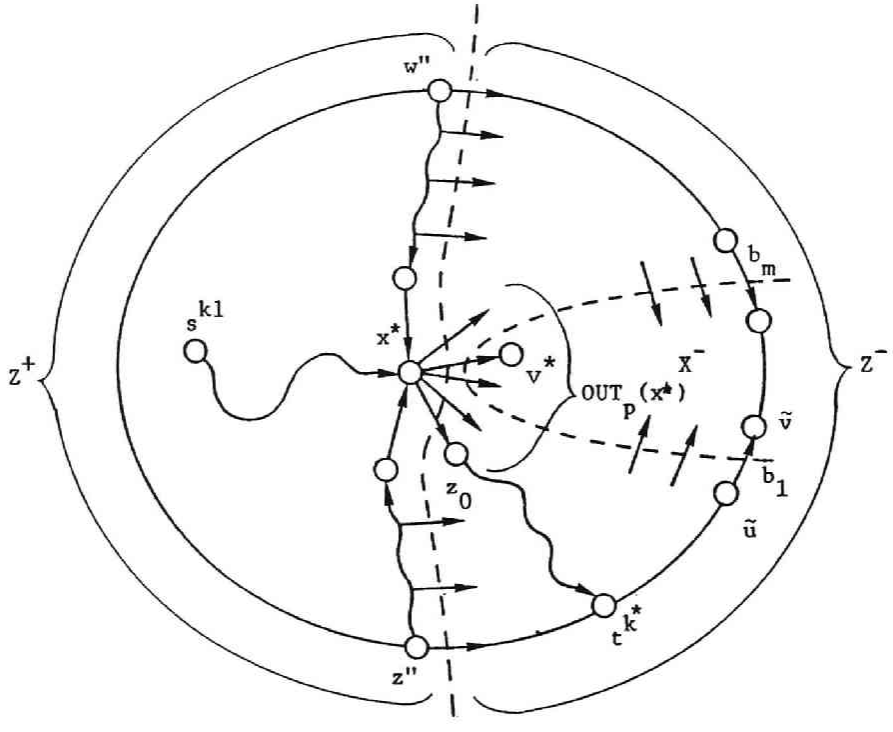


Fig. 6.5 Proof of Lemma 6.2.

Assumption 6.1 (a) By Lemma 6.2, we have $t^{k*} \in (Z^- - X^-) \cap V_B$. The w" and z" defined as above satisfy $t^{k*} \in VB(w",\tilde{u})$ or $t^{k*} \in VB[\tilde{u},z")$, as illustrated in Figure 6.5. In subsequent sections 6.4, 6.5 and 6.6, we assume $t^{k*} \in VB[\tilde{u},z")$ without loss of generality, because the other case $t^{k*} \in VB(w",\tilde{u})$ can be treated similarly by reversing the left-right relation.

(b) (T-4) holds for some $a(x^*,v^*) \in OUT_p(x^*)$. Now we select the arc $a(x^*,v^*)$ such that any arc (in $OUT_p(x^*)$) to the right of $a(x^*,v^*)$ does not satisfy (T-4). That is, the rightmost arc a' in the set of arcs satisfying (T-4) can be chosen as such $a(x^*,v^*)$. (If a pair of arc a' and commodity k', and a pair of arc a" and k" both satisfy (T-4), we see that if a' is to the right of a" then $t^{k'}$ is also to the right of $t^{k"}$ by the rule of flow assignment used in ASSIGN. Therefore this definition of $a(x^*,v^*)$ does not contradict assumption (a)). □

Assumption 6.1 is made throughout this chapter.

Lemma 6.3 In (N,f), these exists an arc $a(x^*,z_0) \in OUT_p(x^*)$ that is located to the right of $a(x^*,v^*)$ and satisfies $\Pi(z_0,t^{k*}) \neq \emptyset$. This is illustrated in Figure 6.5.

Proof. If $x^* \neq s^{k*}$, $k^* \in K(a(u,x^*))$ holds for some $a(u,x^*) \in IN(x^*)$. $\Pi(x^*,t^{k*}) \neq \emptyset$ holds because u is scanned

(i.e., (T-4) is not satisfied). On the other hand, if $x^*=s^{k*}$, we obtain $\Pi(x^*,t^{k*})\neq\emptyset$ immediately from Assumption A. Here since $t^{k*}\in Z^-$ by Lemma 6.2, any path in $\Pi(x^*,t^{k*})$ contains some $a(x^*,z_0)\in OUT_p(x^*)$ and $z_0\notin X^-$ by definition of Z^- and X^-. If such $a(x^*,z_0)$ is to the left of $a(x^*,v^*)$, this contradicts the planarity of N by Assumption 6.1 (a). Therefore $a(x^*,z_0)$ is located to the right of $a(x^*,v^*)$. □

Lemma 6.4 For $v\in DIV(Z^+)$ and $y\in REACH(x^*)$ with $\Pi(v,y)\neq\emptyset$ in (N,f), choose two arcs $a,b\in OUT_j(y)$ such that a is immediately to the left of b, as illustrated in Figure 6.6. Then $TK(a)\subseteq VB(v,\tilde{v}]$ or $TK(b)\subseteq VB[\tilde{u},v)$ hold.

Proof. Since $y\in REACH(x^*)$ has already passed Step 2, all commodities of the flow in $OUT_j(y)$ are contained in $K_j(y)$. Let X_j^- denote a connected convergent set obtained by applying Lemma 2.5 to $OUT_j(y)$. Then $v\notin X_j^-$ by the convergency of X_j^-. ASSIGN has assigned the commodities in $K_j(y)$ in the left-right order to the arcs in $OUT_j(y)$ chosen in the left-right order. Therefore each commodity in b is equal to or to the right of any commodity of flow in a. Therefore, if $TK(a)\cap VB[\tilde{u},v)\neq\emptyset$ holds, then any sink in TK(b) is not to left to any sink $t^{k'}\in TK(a)\cap VB[\tilde{u},v)$, i.e.,

$TK(b) \subseteq VB[\tilde{u},v)$ holds. In case $TK(a) \cap VB[\tilde{u},v) = \emptyset$ holds, we obtain $TK(a) \subseteq VB(v,\tilde{v}]$ immediately. □

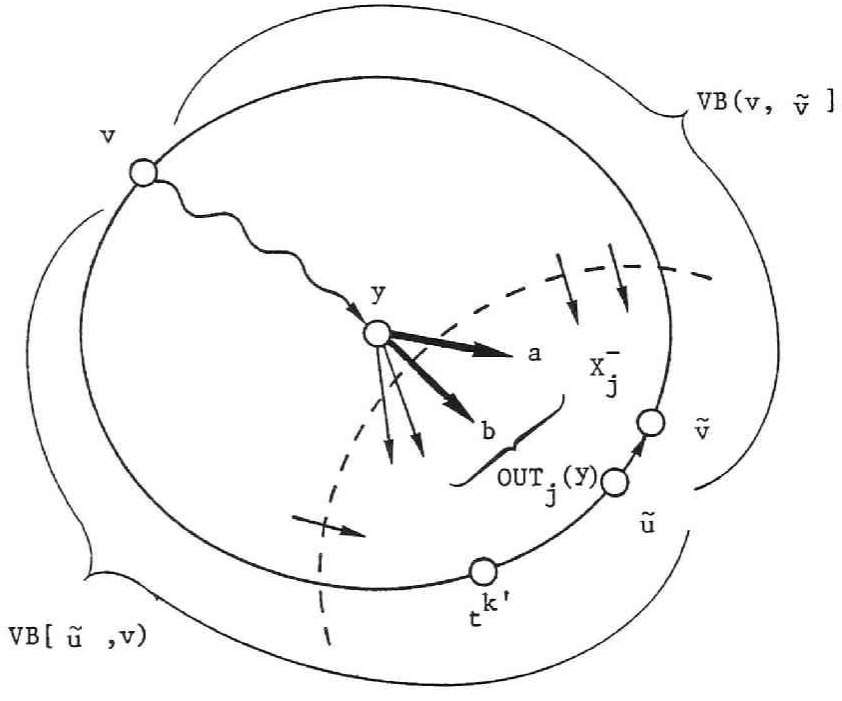


Fig. 6.6 Proof of Lemma 6.4.

In (N,f), take a window that contains an arc already assigned flow. Let W_B denote the window corresponding to the boundary. An arc in W is called <u>forward</u> (<u>backward</u>) arc in W if the arc direction is clockwise (counterclockwise) along W. From acyclicity of G, we see that any window W has at least one forward arc and one backward arc.

Definition 6.3 For a window W ($\neq W_B$) and an arc a in W, we define arcs $a^+(W,a)$ and $a^-(W,a)$ as follows.

(1) If arc a is a backward arc in W, let $a^+(W,a)$ be the first forward arc encountered when we go around from the initial node of a in the clockwise manner along W. Let $a^-(W,a)$ be the backward arc whose initial node is common to $a^+(W,a)$, (possibly $a=a^-(W,a)$). See Figure 6.7.

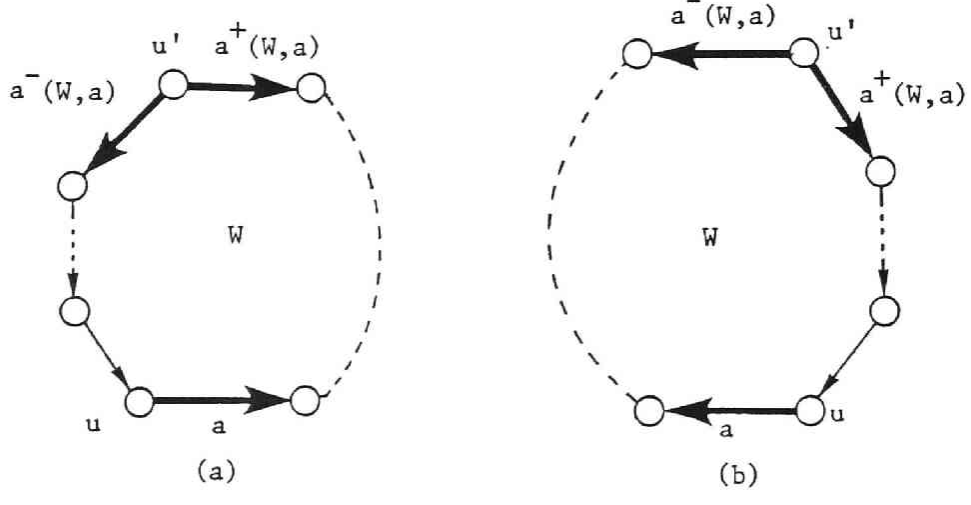


Fig. 6.7 Explanation of Definition 6.3.

(2) If arc a is a forward arc in W, let $a^-(W,a)$ be the first backward arc encountered when we go around from the initial node of a in the counterclockwise manner along W. Let $a^+(W,a)$ be the forward arc whose initial node is common to $a^-(W,a)$, (possibly $a=a^+(W,a)$). See Figure 6.7. □

Consider window W_6 and arc e_{17} shown in Figure 6.4. e_{17} is backward in W_6. Therefore and then $a^+(W_6,e_{17})=e_{11}$ and $a^-(W_6,e_{17}) =e_{12}$.

In each of (a) and (b), if arc a has already been assigned flow by ASSIGN then flow in $a^+(W,a)$ and $a^-(W,a)$ have been determined, because the initial node of $a^+(W,a)$ is reachable to the initial node of arc a.

$DIV(Z^+) \subseteq VB[z",w"]$ was already stated. Let $u_1,u_2 \in VB[z",w"]$. If u_1 is to the left of u_2 in VB[z",w"] (i.e., $u_2 \in VB(u_1,w"]$), this is denoted by $u_1>u_2$. The case $u_1=u_2$ or $u_1>u_2$ is denoted by $u_1 \geq u_1$. For each u REACH(x^*), let $v_R(u)$ and $v_L(u)$ be the rightmost divergent node and the leftmost divergent node, respectively, among these nodes in $DIV(Z^+)$ which are reachable to u. See Figure 6.8.

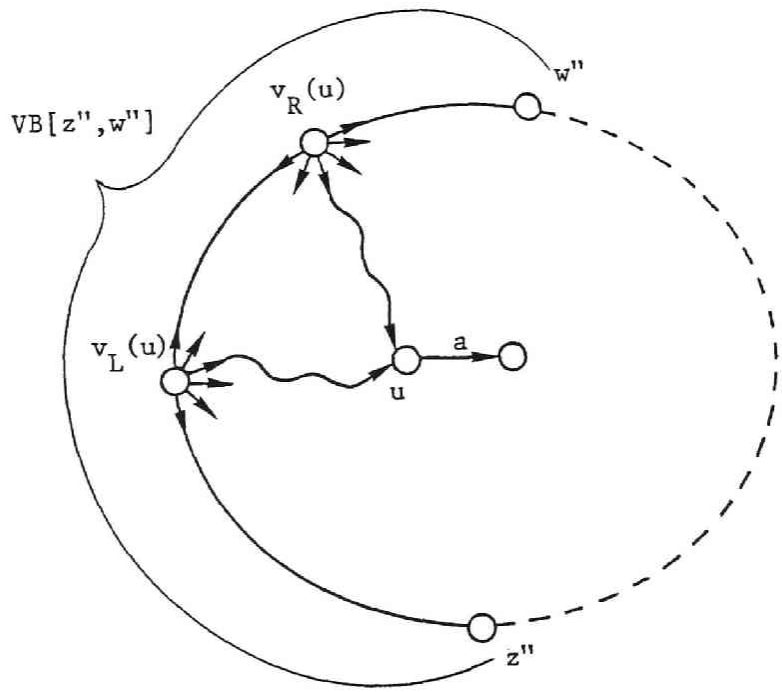


Fig. 6.8 Definition of $v_L(u)$ and $v_R(u)$.

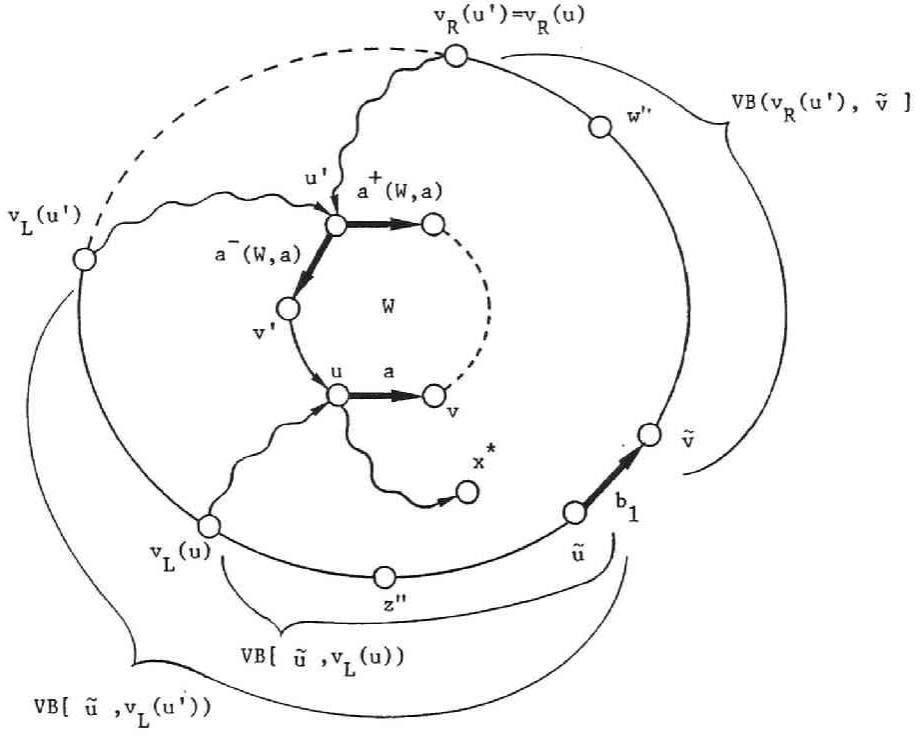


Fig. 6.9 Illustration of Lemma 6.5(1).

Lemma 6.5 Under the above notation, a window W ($\neq W_B$) and an arc a in W satisfy the following, where u and u' are the initial nodes of arc a and arc $a^+(W,a)$, respectively.

(1) If arc a is backward in W, $u \in REACH(x^*)$ and $TK(a) \subseteq VB(v_R(u),\tilde{v}]$, then $u' \in REACH(x^*)$ and $v_L(u) \geq v_L(u') \geq v_R(u') \geq v_R(u)$ hold. Furthermore one of the following (i) and (ii) holds, as illustrated in Figure 6.9.

(i) $TK(a^+(W,a)) \subseteq VB(v_R(u'),\tilde{v}]$ and $a^+(W,a) \neq a$.

(ii) $TK(a^-(W,a)) \subseteq VB[\tilde{u},v_L(u'))$ and $a^-(W,a) \neq a$.

(2) If arc a is forward in W, $u \in REACH(x^*)$ and $TK(a) \subseteq VB[\tilde{u},v_L(u))$, then $u' \in REACH(x^*)$ and $v_L(u) \geq v_L(u') \geq v_R(u') \geq v_R(u)$ hold. Furthermore one of the following (i) and (ii) holds.

(i) $TK(a^+(W,a)) \subseteq VB(v_R(u'),\tilde{v}]$ and $a^+(W,a) \neq a$.

(ii) $TK(a^-(W,a)) \subseteq VB[\tilde{u},v_L(u'))$ and $a^-(W,a) \neq a$.

Proof. We prove only (1), as (2) is similar. First we show $v_L(u) \geq v_L(u') \geq v_R(u') \geq v_R(u)$. Take a divergent node $\hat{v}$ with $\Pi(\hat{v},u') \neq \emptyset$ (i.e., $v_L(u') \geq \hat{v} \geq v_R(u')$). Then, since $v_L(u) \geq \hat{v} \geq v_R(u)$ holds from $\Pi(u',u) \neq \emptyset$, we have $v_L(u) \geq v_L(u') \geq v_R(u') \geq v_R(u)$.

Now we show that (i) or (ii) always holds. For the ternimal node v of arc a, let Y[v] denote the set of nodes $y \in V_B$ with $\Pi(v,y) \neq \emptyset$ (for $v \in V_B$, assume $v \in Y[v]$). Since V_B contains a convergent node which is reachable from $v \notin V_B$, $Y[v] \neq \emptyset$ always holds. Here we consider the following three cases.

Case-1: $Y[v] \cap VB(v_L(u),v_R(u)) \neq \emptyset$, $Y[v]-VB(v_L(u),v_R(u)) \neq \emptyset$,

Case-2: $Y[v] \subseteq VB(v_L(u),v_R(u))$,

Case-3: $Y[v] \cap VB(v_L(u),v_R(u)) = \emptyset$.

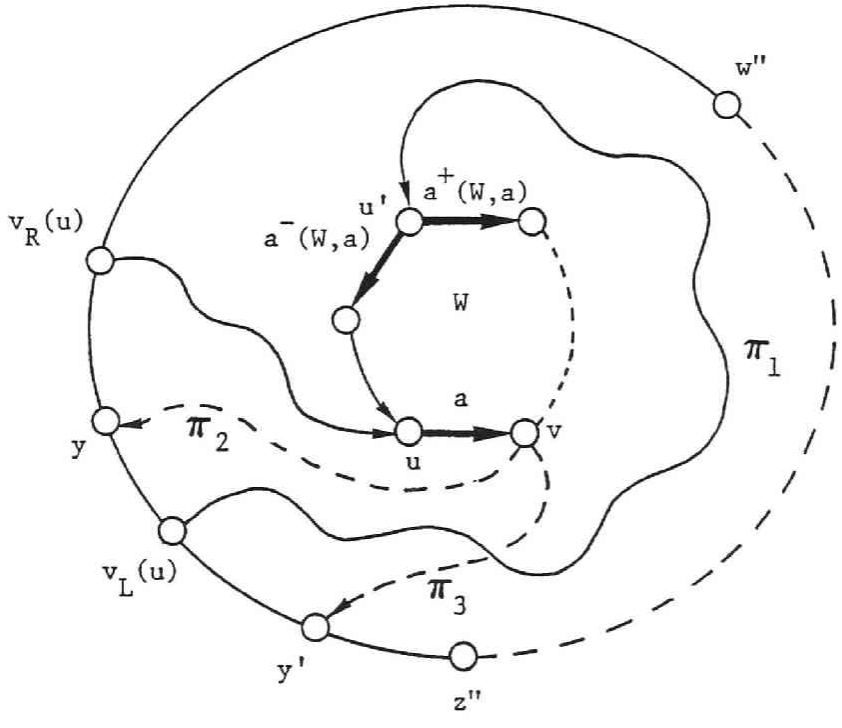


Fig. 6.10 Proof of Lemma 6.5.

In Case-1, we choose y and y' such that $y \in Y[v] \cap VB(v_L(u),v_R(u))$ and $y' \in Y[v]-VB(v_L(u),v_R(u))$, as shown in Figure 6.10. Clearly $v_L(u) \in VB(y',y)$. From the planarity, any path $\pi_1 \in \Pi(v_L(u),u)$ has a node common to either $\pi_2 \in \Pi(v,y)$ or $\pi_3 \in \Pi(v,y')$, contradicting the acyclicity of G. Therefore Case-1 is impossible.

In Case-2, we first show that $TK(a) \subseteq VB(v_L(u),v_R(u))$ holds. Consider the case $u \neq x^*$. This $u \in REACH(x^*)$ has already passed Steps 2 and 3, and therefore $\Pi(v,t^k) \neq \emptyset$ for any sink $t^k \in TK(a)$. That is, we have $TK(a) \subseteq Y[v] \subseteq VB(v_L(u),v_R(u))$. On the contrary, if $u=x^*$, x^* has passed Step 2, and therefore all sinks of the commodities in flow in $OUT_j(x^*)$ (containing arc a) are located in $\bar{X}_j$, which is obtained by Lemma 2.5. See Figure 6.11. From $u=x^* \notin \bar{X}_j$ and convergency of $\bar{X}_j$, set $\bar{X}_j$ contains no node reachable to x. Hence, $v_L(u),v_R(u) \notin \bar{X}_j$. From $Y[v] \subseteq \bar{X}_j$ and the assumption of Case-2, we have $\bar{X}_j \cap VB(v_L(u),v_R(u)) \neq \emptyset$, that is, $(TK(a) \subseteq) \bar{X}_j \subseteq VB(v_L(u),v_R(u))$. The result $TK(a) \subset VB(v_L(u),v_R(u))$ however means $VB(v_L(u),v_R(u)) \cap VB(v_R(u),\tilde{v}] = \emptyset$ because $\tilde{v} \notin VB(v_L(u),v_R(u))$ holds by definition of $\tilde{v}$, where $\tilde{v}$ is the terminal node of b_1 in Definition 6.2. This contradicts the condition of the lemma, $TK(a) \subseteq VB(v_R(u),\tilde{v}]$. Therefore Case-2 is also

impossible.

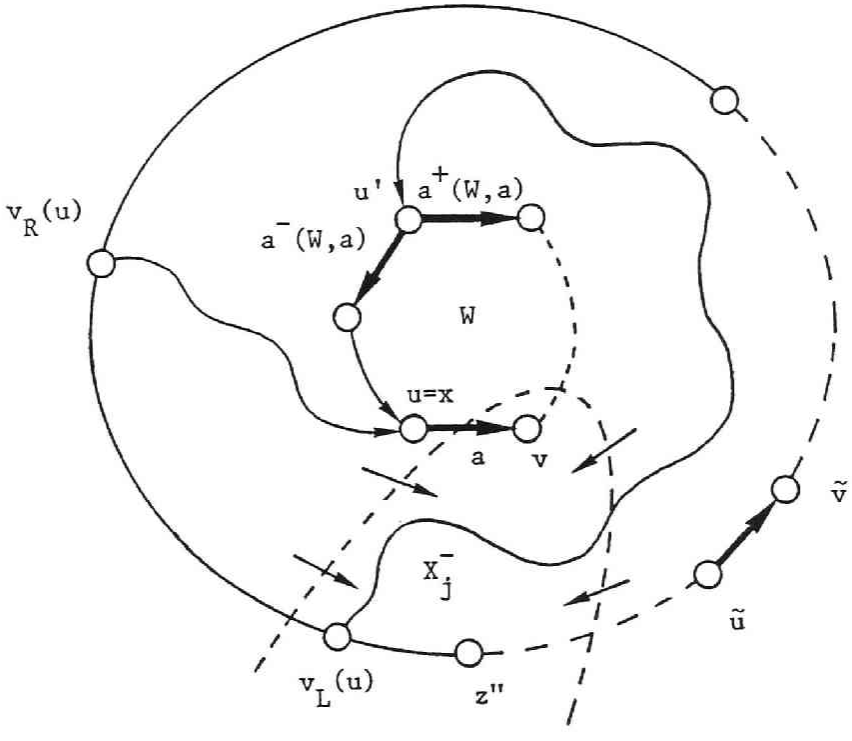


Fig. 6.11 Proof of Lemma 6.5.

In Case-3, by applying Lemma 6.4 to $a=a^{+}(W,a)$, $b=a^{-}(W,a)$, $y=u'$ and $v=v_R(u)$, we have $TK(a^{+}(W,a)) \subseteq VB(v_R(u'),\tilde{v}]$ or $TK(a^{-}(W,a)) \subseteq VB[\tilde{u},v_R(u'))$. Since the former satisfies the lemma's statement (obviously backward arc a differs from $a^{+}(W,a)$), we assume the latter. For the terminal node v' of arc $a^{-}(W,a)$, let Y[v'] denote the set of nodes $y \in V_B$ with $\Pi(v',y)\neq\emptyset$. For $v' \in V_B$, let $v' \in Y[v]$. Hence, $Y[v] \subseteq Y[v']\neq\emptyset$ from $\Pi(v',v)\neq\emptyset$, and $Y[v] \cap VB(v_L(u'),v_R(u'))=\emptyset$ from the assumption $Y[v] \cap VB(v_L(u),v_R(u))=\emptyset$. Here if we assume $Y[v'] \cap VB(v_L(u'),v_R(u'))\neq\emptyset$, then $Y[v']-VB(v_L(u'),v_R(u')) \supseteq Y[v]\neq\emptyset$. In this case, since $Y[v'] \cap VB(v_L(u'),v_R(u'))\neq\emptyset$ and $Y[v']-VB(v_L(u'),v_R(u'))\neq\emptyset$, we can derive a contradiction from the acyclicity of G in a manner similar to Case-1. Therefore we obtain $Y[v'] \cap VB(v_L(u'),v_R(u'))=\emptyset$. See Figure 6.12. This u' is satisfies $u' \in REACH(x^{*})$ by $u \in REACH(x^{*})$ and $\Pi(u',u)\neq\emptyset$. Furthermore $u'\neq x^{*}$ can be shown as follows. If $u=x^{*}$ holds, $u'\neq x^{*}$ is obvious from $u'\neq u$. If $u\neq x^{*}$ holds, we have $\Pi(u',u)\neq\emptyset$ and hence $u'\neq x^{*}$ by the acyclicity of G and $\Pi(u,x^{*})\neq\emptyset$. Therefore $(x^{*}\neq)u' \in REACH(x^{*})$ has passed Steps 2 and 3 of ASSIGN, and $\Pi(v',t^{k})\neq\emptyset$ holds for any $t^{k} \in TK(a^{-}(W,a))$. Then we obtain

$TK(a^-(W,a)) \subseteq Y[v'] \subseteq V_B - VB(v_L(u'), v_R(u'))$. Therefore, the assumption $TK(a^-(W,a)) \subseteq VB[\tilde{u}, v_R(u'))$ implies $TK(a^-(W,a)) \subseteq VB[\tilde{u}, v_R(u'))$. Consequently we have $TK(a^-(W,a)) \cap TK(a) = \emptyset$ (i.e., $a^-(W,a) \neq a$) by the condition of the lemma, $TK(a) \subseteq VB(v_R(u), \tilde{v}]$. Hence (ii) holds. □

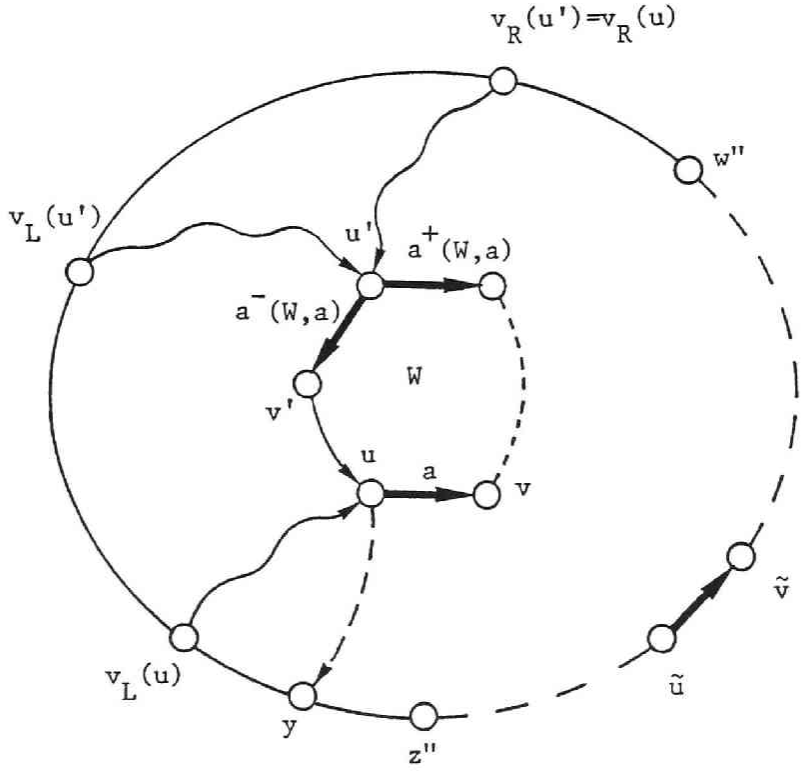


Fig. 6.12 Proof of Lemma 6.5.

In (N,f), let W'_i denote the window containing arcs b_i and b_{i+1} (b_1 and b_m are the arcs contained in W_B), where $\{b_1,b_2,\ldots,b_m\}$ is the unilateral simple cut $A(V-X^-)$ in Definition 6.2. There exists the r-th arc b_r in $A(V-X^-)$ such that the initial node of b_r differs from x^* and the initial node of b_{r+1} is equal to x^*, as shown in Figure 6.13.

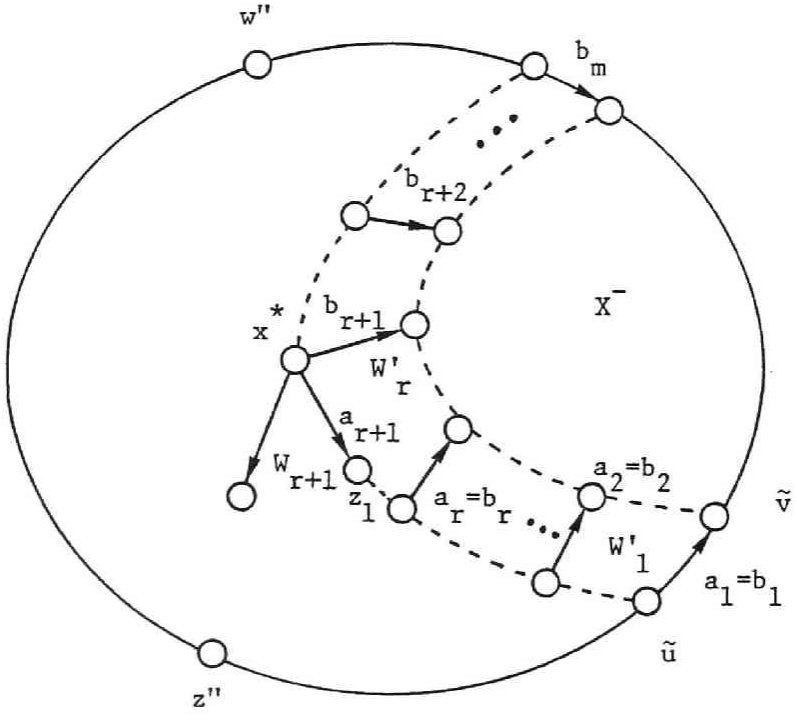


Fig. 6.13 Illustration of arc a_r.

In Figure 6.4, we obtain $b_r=b_2=a(y_{14},y_{15})$ and $b_{r+1}=b_3=e_{23}$.

Lemma 6.6 Let $b_i=a(u_i',v_i')$, i=1,2,...,r. Then any u'_i satisfies $u_i' \notin$ REACH(x^*), i.e., flows in b_i(i=1,2,...,r) are not determined yet in (N,f).

Proof. Assume that $u_1' \in$ REACH(x^*) holds for some $b_i(1\leq i\leq r)$. Then a simple path $\pi_1 \in \Pi(u_i',x^*)$ exists. Here we consider the graph obtained by shrinking the connected convergent set X^- into a convergent node x^-, as shown in Figure 6.14. By Lemma 2.6, G' is also planar and acyclic. In G', $a(x^*,x^-)$,b_i and π_1 form a simple undirected cycle C. In G', $a(x^*,z_0)$ is located to the right of $a(x^*,x^-)$, and therefore $a(x^*,z_0)$ is contained in the area surrounded by C, where $a(x^*,z_0)$ is the arc defined in Lemma 6.3 such that $\Pi(z_0,t^{k*})\neq\emptyset$ and $a(x^*,z_0) \in OUT_p(x^*)$. From $t^{k*} \in V_B$, $t^{k*}(\neq x^-)$ is located in C or in the outside of the area surrounded by C. This means that π_1 and $\pi_2 \in \Pi(z_0,t^{k*})$ consist of a directed cycle, contradicting the acyclicity of G'. □

Lemma 6.7 For the index r in Lemma 6.6, W_r' contains a node z_1 such that $b_{r+1}\neq a(x^*,z_1) \in A$ and $TK(a(x^*,z_1)) \subseteq VB[t^{k*},z'')$ hold. This is illustrated in Figure 6.13.

Proof. If $OUT_p(x^*)$ contains no arc located to the

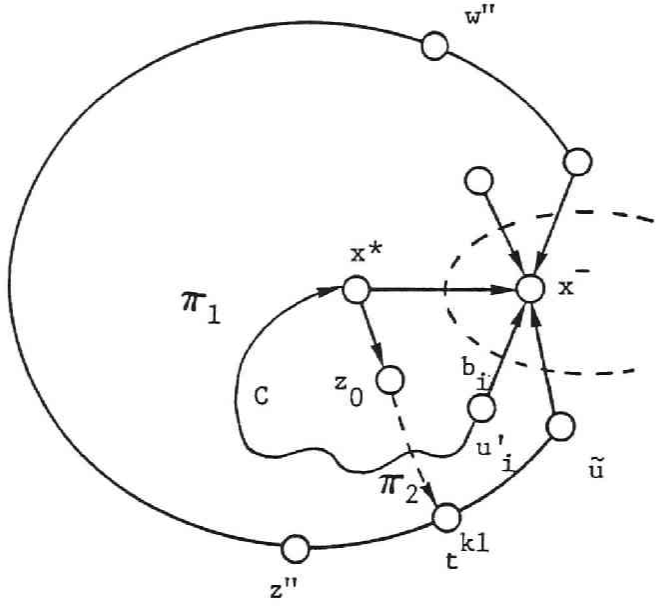


Fig. 6.14 Proof of Lemma 6.7

right of b_{r+1}, then this contradicts the existence of $a(x^*,z_0)$ satisfying Lemma 6.3. Therefore we can choose an arc $a(x^*,z_1)$ located to the right of b_{r+1}. Clearly $a(x^*,z_1)$ is contained in W_r'. Since x^* has passed Step 2 of ASSIGN, flow has already been assigned to each arc in $OUT_p(x^*)$ and $TK(a) \subseteq Z^- \cap V_B$ holds for each $a \in OUT_p(x^*)$. Now consider $TK(a(x^*,z_1))$. As $a(x^*,z_1)$ is to the right of $a(x^*,v^*)$ in

Assumption 6.1, $t^{k*} \in TK(a(x^*,v^*))$ means that any commodity of flow in $a(x^*,z_1)$ is to the right of k^*. That is, $TK(a(x^*,z_1) \subseteq VB[t^{k*},z")$ holds. □

In Figure 6.4, we obtain $W'_r=W_2$, $a(x^*,z_1)=e_{24}$ and $z_1=y_{10}$.

6.4 ALGORITHM FIND-CUT

Assume that ASSIGN has found $a(x^*,v^*) \in OUT_p(x^*)$ satisfying (T-4) and then halted. Given the (N,f) of Definition 6.1, the following procedure constructs a simple cut C^* not satisfying the cut-condition (2,1). In this section, after presenting the procedure, we clarify some properties.

Procedure FIND-CUT

0. For the unilateral simple cut $A(V-X^-)=\{b_1,b_2,\ldots,b_m\}$ constructed in Section 6.3, obtain the set of arcs $\{b_1,b_2,\ldots,b_r\}$ defined prior to Lemma 6.6. Let $a_i:=b_i$ $(i=1,2,\ldots,r)$. Let $a_{r+1}:=a(x^*,z_1)$, where $a(x^*,z_1)$ is the arc satisfying Lemma 6.7. Based on window W_B and windows W_i containing a_i and a_{i+1} (i.e., equal to those W_i' defined prior to Lemma 6.6) $(i=1,2,\ldots,r)$, we define a set of windows $Q=\{W_0,W_1,W_2,\ldots,W_r\}$. Further let $C^-:=\{a_1,a_2,\ldots,a_r\}$, $C^+:=\{a_{r+1}\}$ and $i:=r+1$.

1. Let W_i be the other window (distinct from W_{i-1}) containing a_i. If $W_i \in Q$ then proceed to 5. Otherwise, let $Q:=Q \cup \{W_i\}$ and proceed to 2.

2. If $a_i \in C^-$ (i.e., a_i is a backward arc in W_i), the initial node u of a_i satisfies $u \in REACH(x^*)$ and

$TK(a) \subseteq VB(v_R(u), \tilde{v}]$. In this case, (i) or (ii) of Lemma 6.5 (1) holds. (The validity is proved in the following Lemma 6.10.)

If $a_i \in C^+$ (i.e., a_i is a forward arc in W_i), the initial node u of a_i satisfies $u \in REACH(x^*)$ and $TK(a) \subseteq VB[\tilde{u}, v_L(u))$. In this case, (i) or (ii) of Lemma 6.5 (2) holds, i.e.,

(i) $TK(a^+(W_i,a_i)) \subseteq VB(v_R(u'), \tilde{v}]$ and $a^+(W_i,a_i) \neq a_i$,

(ii) $TK(a^-(W_i,a_i)) \subseteq VB[\tilde{u}, v_L(u'))$ and $a^-(W_i,a_i) \neq a_i$,

where u' is the node common to $a^+(W_i,a_i)$ and $a^-(W_i,a_i)$. Proceed to 3 if (i) holds. Otherwise, proceed to 4.

3. Let $a_{i+1} := a^+(W_i,a_i)$, $C^- := C^- \cup \{a_{i+1}\}$ (a_{i+1} is a backward arc in W_{i+1}) and $i := i+1$. Return to 1.

4. Let $a_{i+1} := a^-(W_i,a_i)$, $C^- := C^- \cup \{a_{i+1}\}$ (a_{i+1} is a forward arc in W_{i+1}) and $i := i+1$. Return to 1.

5. Let $C^* := C^+ \cup C^-$ and halt. This situation is illustrated in Figure 6.15. □

Since the number of updating $Q := Q \cup \{W_i\}$ in Step 1 of FIND-CUT is at most the number of all windows in G, FIND-CUT halts after finite number of operations. Here we estimate the time complexity of FIND-CUT. To obtain $a^+(W_i,a_i)$ and $a^-(W_i,a_i)$ for some a_i in Step 2, we go around the cycle W_i

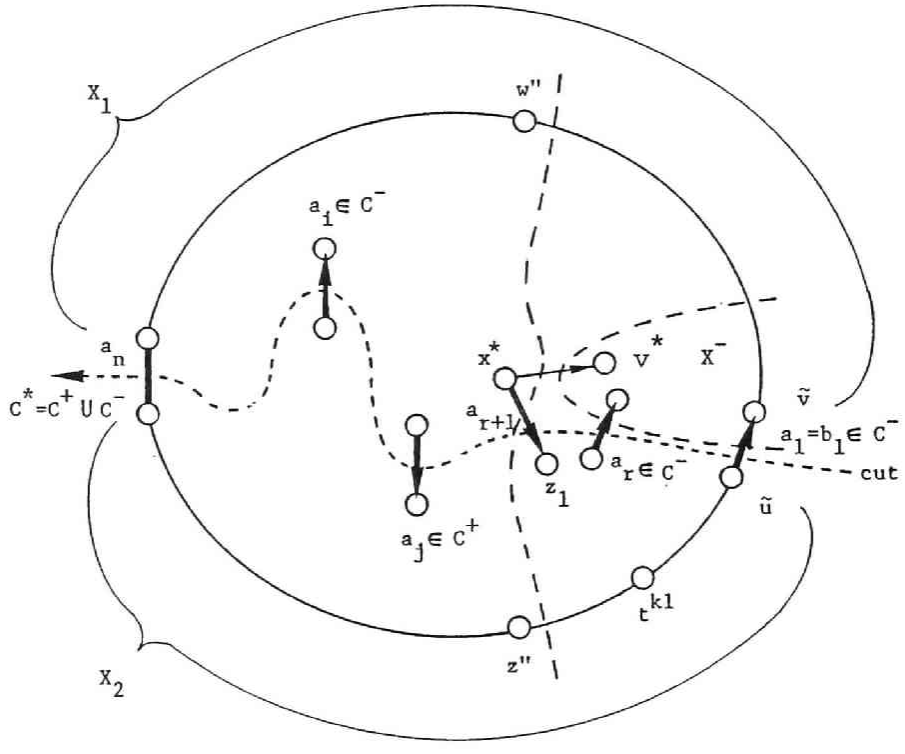


Fig. 6.15 Cut C^* obtained by algorithm FIND-CUT.

from the initial node u_i of a_i to the node u' common to $a^+(W_i,a_i)$ and $a^-(W_i,a_i)$. Since FIND-CUT does not search a window more than twice and furthermore an arc is contained in exactly two distinct windows, any arc is not searched by FIND-CUT more than three times. Therefore, the total of the time required to obtain $a^+(W_i,a_i)$ and $a^-(W_i,a_i)$ for all i is bounded from above by $O(|A|)$ (i.e., $O(|V|)$ by the planarity). Now we consider the time required to determine whether (i) holds or not in Step 2. Here we make use of the preprocessing by which $v_R(u')$ and $VB(v_R(u'),\tilde{v}] \cap T$ are computed for each $u' \in REACH(x^*)$. That is, to obtain $v_R(u')$ for each $u' \in REACH(x)$, we scan a divergent node (say $\hat{v}$) from right to left among VB[z",w"] and set $v_R(u')=\hat{v}$ for all $u' \in REACH(x^*)$ such that u' is reachable from $\hat{v}$ but $v_R(u')$ is not yet set. In this process, no arc is searched more than twice, and therefore the required time is bounded from above by $O(|A|+|V|)$, i.e., $O(|V|)$. Also the time required to obtain $VB(v_R(u'),\tilde{v}] \cap T$ from the set of all $v_R(u')$ is bounded from above by $O(K|V|)$, because the set of the searched nodes is located only in the boundary. Based on the data obtained by preprocessing, we can determine whether (i) of Step 2 holds or not in $O(|K|)$ time. Since the number of iterations in Step 2 is $O(|Q|)(=O(|V|))$, the total time of

FIND-CUT and the preprocessing is bounded from above by $O(K|V|)$.

Applying FIND-CUT to the network in Figure 6.4, we obtain

$$v^*=\tilde{v}=y_{15},\ \tilde{u}=y_{11},\ t^{k*}=y_{12},\ z_0=z_1=y_{10},$$

$$a_1=b_1=a(y_{11},y_{15}),\ a_2=b_2=b_r=a(y_{14},y_{15}),\ a_3=e_{24},$$

$$a_4=e_{25},\ a_5=e_{18},\ a_6=e_{17},\ a_7=a_n=e_{11},$$

$$C^+=\{a_3,a_4\},\ C^-=\{a_1,a_2,a_5,a_6,a_7\},$$

$$X_1=\{y_1,y_2,y_3,y_7,y_8,y_{15},y_{16},\bar{x_1}(=\{y_{17},y_{18}\})\},$$

$$X_2=\{y_4,y_5,y_6,y_9,y_{10},y_{11},y_{12},y_{13},y_{14}\},$$

where X_1 and X_2 are the sets of nodes obtained after removing cut $C^*=C^+\cup C^-$. Windows W_i, $i=1,2,\ldots,6$ are shown in Figure 6.4. Here,

$$c(X_1)=c(e_{24})+c(e_{25})=f(e_{24},6)+f(e_{25},6)=4$$

holds. However, $f(e_{23},5)=1>0$ for $a(x^*,v^*)=e_{23}$, and hence

$$r(X_1)=c(X_1)-g(X_1)\leq c(X_1)-(g^5+g^6)=4-(4+1)<0$$

holds, indicating that C^* does not satisfy the cut-condition (2.1).

The validity of Step 2 of FIND-CUT is shown by the following lemma.

Lemma 6.8 Let $a_i=a(u_i,v_i)$ denote the a_i $(i\geq r+1)$ obtained by FIND-CUT. Then $TK(a_i)\neq\emptyset$ and $u_i\in REACH(x^*)$ always hold.

Furthermore, $TK(a_i) \subseteq VB(v_R(u_i),\tilde{v}]$ holds if a_i is a backward arc in W_i (i.e., $a_i \in C^-$), and $TK(a_i) \subseteq VB[\tilde{u},v_L(u_i))$ holds if a_i is a forward arc in W_i (i.e., $a_i \in C^+$).

Proof. We show by induction on i ($i \geq r+1$).

(I) For $i=r+1$, we have $a_{r+1}=a(x^*,z_1) \in C^+$ and $u_{r+1}=x^* \in REACH(x^*)$. Here $TK(a(x^*,z_1)) \subseteq VB[t^{k*},z'')$ by Lemma 6.7, and then $TK(a_{r+1}) \subseteq VB[t^{k*},z'') \subseteq VB[\tilde{u},z'')$ by $t^{k*} \in VB[\tilde{u},z'')$ of Assumption 6.1 (a). Furthermor, since $v_L(u_{r+1}) \in VB[z'',w']$ from the definition of v_L, we have $TK(a_{r+1}) \subseteq VB[\tilde{u},z'') \subseteq VB[\tilde{u},v_L(u_{r+1})]$.

(II) For $i=k$ ($k \geq r+1$), we assume that the lemma holds (i.e., the condition of Lemma 6.5 holds). Since a_{k+1} is equal to $a^+(W_k,a_k)$ or $a^-(W_k,a_k)$, $\Pi(u_{k+1},u_k) \neq \emptyset$ by Definition 6.3. Here a_k satisfies the condition of Lemma 6.5, and then we have $u_k \in REACH(x^*)$ and furthermore $u_{k+1} \in REACH(x^*)$ (i.e., $TK(a_{k+1}) \neq \emptyset$ by Lemma 6.1). If (i) of Lemma 6.5 holds for $i=k$, then $a_{k+1}=a^+(W_k,a_k)$ implies that $a_{k+1} \in C^-$ and $TK(a^+(W_k,a_k)) \subseteq VB(v_R(u_{k+1}),\tilde{v}]$. If (ii) of Lemma 6.5 holds for $i=k$, then $a_{k+1}=a^-(W_k,a_k)$ implies that $a_{k+1} \in C^+$ and $TK(a^-(W_k,a_k)) \subseteq VB[\tilde{u},v_L(u_{k+1}))$. □

Lemma 6.9 Let C^- and C^+ be the sets of arcs obtained by FIND-CUT. Then $TK(a_i) \cap TK(a_j)=\emptyset$ always holds for $a_i \in C^-$

and $a_j \in C^+$.

Proof. For the set of arcs $C^*=C^+ \cup C^-=\{a_1,a_2,\ldots,a_n\}$, let $a_i=a(u_i,v_i)$, $i=1,2,..,n$. Lemma 6.6 and Lemma 6.8 assert that

$a_i \in C^-$ and $TK(a_i) = \emptyset$ for $i \leq r$,

$\emptyset \neq TK(a_i) \subseteq VB[\tilde{u},v_L(u_i))$ if $i \geq r+1$ and $a_i \in C^+$,

$\emptyset \neq TK(a_i) \subseteq VB(v_R(u_i),\tilde{v}]$ if $i \geq r+1$ and $a_i \in C^-$.

Therefore, to prove the lemma, we show $VB(v_R(u_i),\tilde{v}]$ and $VB[\tilde{u},v_L(u_j))$ are always disjoint for $a_i \in C^+$ and $a_j \in C^-$ with $i,j \geq r+1$. That is, we shall prove that $v_L(u_j) \geq v_R(u_i)$ holds.

By Lemma 6.5 and the way of constructing a_i, we have

$$v_L(u_i) \geq v_L(u_{i+1}) \geq v_R(u_{i+1}) \geq v_R(u_i) \quad (r+1 \leq i \leq n-1). \tag{6.4}$$

Hence, for $r+1 \leq p < q \leq n-1$, we obtain

$$v_L(u_p) \geq v_L(u_q) \text{ and } v_R(u_q) \geq v_R(u_p). \tag{6.5}$$

Assume that $v_L(u_j) < v_R(u_i)$ holds for some i and j. From the definition of $v_R(u)$ and $v_L(u)$, $v_R(u_i) \leq v_L(u_i)$ always holds. As $i \neq j$ holds obviously, we have $v_R(u_i) > v_L(u_j) \geq v_R(u_j)$ for $j<i$. This, however, contradicts the latter condition of (6.5). Otherwise, $v_L(u_j) \geq v_R(u_i) > v_L(u_j)$ holds for $j<i$, contradicting the former condition of (6.5). Therefore we obtain $v_L(u_j) \geq v_R(u_i)$. □

Lemma 6.10 Let C^* be the set of arcs obtained by FIND-CUT. Then C^* contains no arc located to the left of $a(x^*,z_1)$ in $OUT_p(x^*)$.

Proof. From Step 0 of FIND-CUT, $a(x^*,z_1)=a_{r+1} \in C^+$ holds. Assume that $C^+ \cup C^-$ contains some arc $a_q \in OUT_p(x^*)$ to the left of a_{r+1}. Clearly $q>r+1$. By the proof of Lemma 6.8, $\Pi(u_j,u_i)\neq\emptyset$ holds for any pair of i and j with $r+1\leq i<j\leq q$. Here we show $u_i=x^*$ for all i with $r+1\leq i\leq q$. If $u_i\neq x^*$ for some i with $r+1<i<q$, then $u_{r+1}=u_q=x^*$ implies that $\Pi(u_q,u_i)\neq\emptyset$ and $\Pi(u_i,u_{r+1})\neq\emptyset$. That is, there exists a directed cycle containing x and u_i, contradicting the acyclicity. Then we have $u_i=x^*$ for i with $r+1\leq i\leq q$. Window W_i containing a_i and a_{i+1} satisfies $W_i\neq W_B$ for i with $r+1\leq i<q$, because FIND-CUT has constructed Q for i=q. This means that a_i and a_{i+1} for $r+1\leq i<q$ are immediate neighbour each other and a_{i+1} is to the right of a_i. This contradicts the assumption that a_q is to the left of a_{r+1}. □

By the simplicity of cut $A(V-X^-)$, all windows W_i $(0\leq i\leq r)$ in Q at Step 0 are distinct each other. By the termination condition in Step 1, no window is searched by FIND-CUT more than tiwce. Assume that the termination condition holds for $W_n \in Q$ with i=n and FIND-CUT has

halted. Note that this is the first time when the set of windows in Q contains a simlpe cycle in the graph dual to G, because a simple cut in a planar graph corresponds to a simple cycle in the dual graph, as described in Lemma 2.2. This means that the set of arcs $C^*=\{a_1,a_2,\ldots,a_n\}$ obtained by FIND-CUT contains exactly one simple cut (say C'). Since a simple cut can be written as a cutting circiut by Lemma 2.2, this simple cut C' should be represented as $C'=\{a_q,a_{q+1},\ldots,a_n\}$ for some q with $1\leq q\leq n$ and $W_n=W_{q-1}$. Let X_1 and X_2 be the partitions obtained by removing C', where $A(X_1)\subseteq C^+$ and $A(X_2)\subseteq C^-$.

Here we define by $\Pi^k(v,y)$ the set of flow paths of commodity k in (N,f), as follows.

$$\Pi^k(v,y)\triangleq\{\pi\in\Pi(v,y)\mid f(a,k)>0 \text{ for all } a\in A(\pi)\}.$$

Lemma 6.11 Let a(u,v) be an arc in (N,f), where u is being scanned by ASSIGN.

(a) If $t^k\in TK(a(u,v))$ holds for some commodity k with $s^k\in REACH(x^*)$ and $t^k\in Z^+$, then $\Pi^k(v,t^k)\neq\emptyset$.

(b) If $t^k\in TK(a(u,v))$ holds for some commodity k with $s^k\in REACH(x^*)$ and $t^k\in Z^-$, then $\Pi^k(v,z^*)\neq\emptyset$ for some node $z^*\in Z^-$.

Proof. (a) Note that the connected convergent sets in

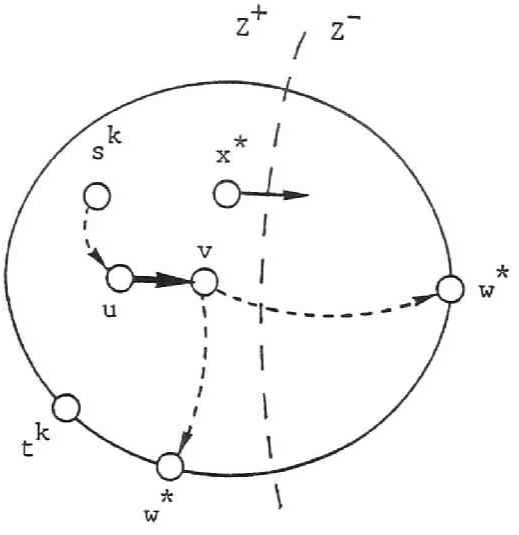


Fig. 6.16 Proof of Lemma 6.11 (a).

V-REACH(x^*) are already shrunk except the set Z^- of Definition 6.1. Assume that $\Pi^k(v,t^k)=\emptyset$ holds for v and t^k satisfying (a) of the lemma. Since flow of commodity k from source s^k satisfies the flow conservation at any node in REACH(x^*), this flow reaches a convergent node w^* ($\neq t^k$) or a node w^* ($\neq x^* \notin Z^-$) in Z^-, as shown in Figure 6.16. This, however, means that $\Pi(w^*,t^k)=\emptyset$ and then ASSIGN must halt by (T-4) before x^* is scanned. This is a contradiction.

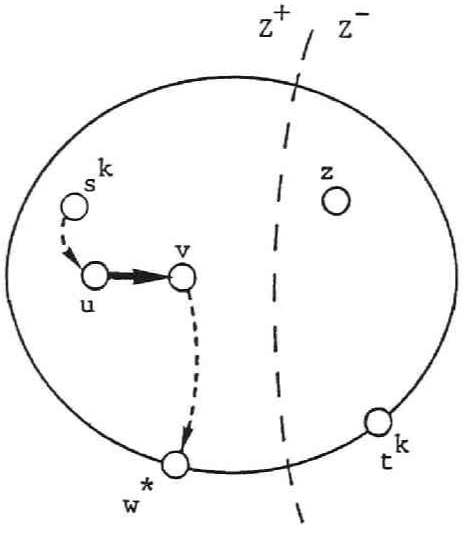


Fig. 6.17 Proof of Lemma 6.11 (b).

(b) Assume that $\Pi^k(v,z)=\emptyset$ holds for all $z \in Z^-$. Since the flow of commodity k from source s^k satisfies the flow conservation at any node in REACH(x^*), $\Pi^k(v,w^*)\neq\emptyset$ holds for some convergent node $w^* \in Z^+ \cap V_B$. See Figure 6.17. Since w^* (or a subgraph shrunk into w^*) does not contain t^k, ASSIGN must halt by (T-4) before x^* is scanned. This is a contradiction. □

Lemma 6.12 Let $C^*=C^+\cup C^-=\{a_1,a_2,\ldots,a_n\}$ be the set of arcs obtained by FIND-CUT, and let $C'=\{a_q,a_{q-1},\ldots,a_n\}$ be the simple cut contained in C^*. Then $q=1$ holds (i.e., $C'=C^*$).

Proof. Since Q contains $W_0(=W_B)$ in Step 0, every a_i $(2\leq i\leq n-1)$ is not contained in W_B. (Otherwise FIND-CUT must have halted earlier.) Since all windows in Q are always distinct each other, $C'\cap A_B\neq\emptyset$ implies that $|C'\cap A_B|=2$, i.e., $C'\cap A_B=\{a_1,a_n\}$, and hence $q=1$ holds. In the subsequent discussion, therefore, we will derive a contradiction under the assumption that $C'\cap A_B=\emptyset$ and $q\geq 2$. This assumption implies that $X_1\cap V_B=\emptyset$ or $X_2\cap V_B=\emptyset$ holds, where the partitions into X_1 and $X_2(=V-X_1)$ obtained by removing C' satisfies as $A(X_1)\subseteq C^+$ and $A(X_2)\subseteq C^-$. See Figure 6.15. We consider the following three cases.

(i) Case of $2\leq q\leq r$ and $X_1\cap V_B=\emptyset$. Let $a_q=a(u_q,v_q)$. Then $u_q\in Z^-$ and $v_q\in X_1$ hold. As all convergent nodes are contained in $V_B\subseteq X_2$, node v_q is not convergent. By the acyclicity of G, there exists a convergent node $w^*\in V_B\subseteq X_2$ with $\Pi(v_q,w^*)\neq\emptyset$. As cut C' separates v_q and w^* as shown in Figure 6.18, C' and $\pi\in\Pi(v_q,w^*)$ have a common arc (say a_j). By $a_j\in A(X_1)$, $a_j\in C^+$ holds. Since $u_q\notin \mathrm{REACH}(x^*)$ holds by Lemma 6.6, $u_j\notin \mathrm{REACH}(x^*)$ holds for the initial

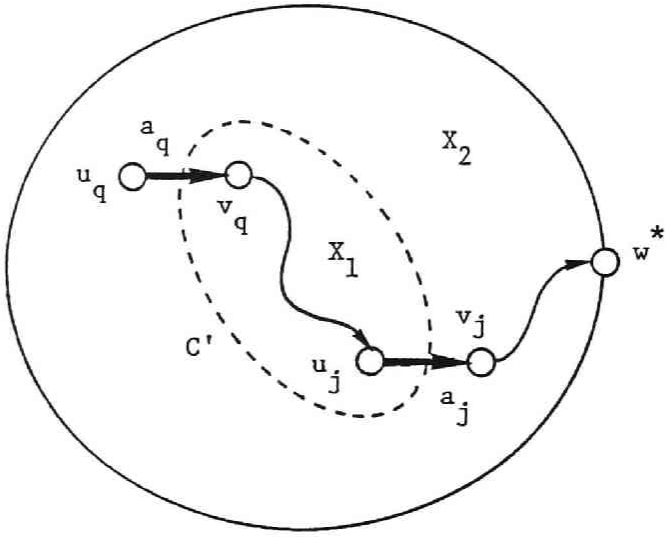


Fig. 6.18 Proof (i) of Lemma 6.12.

node u_j of arc a_j. This arc $a_j \in C^+$, however, satisfies $j \geq r+1$ and this contradicts Lemma 6.8.

(ii) Case of $2 \leq q \leq r$ and $X_2 \cap V_B = \emptyset$. Since W_i $(0 \leq i \leq r)$ are all distinct by the simplicity of cut $A(V-X^-)$, $n \geq r+1$ holds for $C'=\{a_q, a_{q+1}, \ldots, a_n\}$, i.e., $a_{r+1}=a(x^*, z_1) \in C' \cap C^+$, where $z_1 \in Z^-$ is the node in Lemma 6.7. Clearly $x^* \in X_1$ and $z_1 \in X_2$ hold. Since all convergent nodes are contained in $V_B \subseteq X_1$, node z_1 is not convergent. By acyclicity, there exists a convergent node $w^* \in V_B \subseteq X_1$ with $\Pi(z_1, w^*) \neq \emptyset$. Since cut C' separates z_1 and w^*, C' and $\pi \in \Pi(z_1, w^*)$ have a common arc

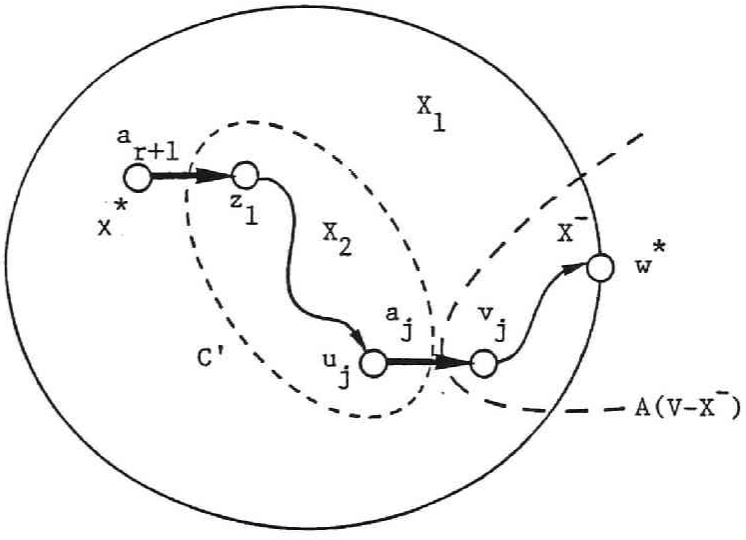


Fig. 6.19 Proof of (ii) of Lemma 6.12.

(say a_j), as shown in Figure 6.19. By $a_j \in A(X_2)$, $a_j \in C^-$ holds, and $\Pi(z_1,x)=\emptyset$ by $z_1 \in Z^-$. Then $\Pi(u_j,x)=\emptyset$ holds for the initial node u_j of a_j. Hence, $u_j \notin REACH(x^*)$ holds and we have $TK(a_j)=\emptyset$ by Lemma 6.1. By Step 0 of FIND-CUT, $j \leq r$ holds if $a_j \in C^-$ satisfies $TK(a_j)=\emptyset$. (That is, $a_j=b_j \in A(V-X^-)$.) Therefore, without using an arc in $A(V-X^-)$, node z_1 cannot be reachable to any sink. Since $TK(a(x^*,z_1)) \subseteq VB[t^{k*},z'') \subseteq V-X^-$ by Lemma 6.7, $a(x^*,z_1)$ satisfies (T-4). This, however, contradicts Assumption 6.1 (b), because $a(x^*,z_1)$ is to the right of $a(x^*,v^*)$.

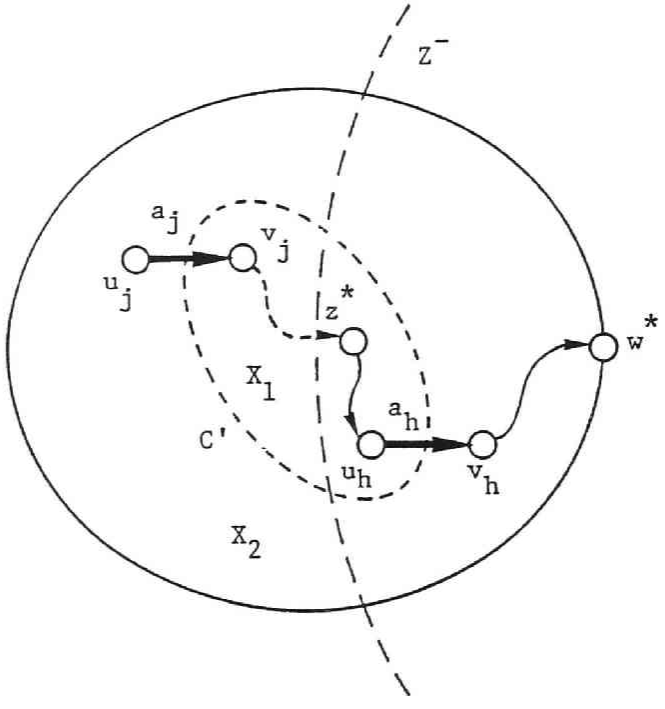


Fig. 6.20 Proof (iii) of Lemma 6.12.

(iii) Case of $r<q\leq n$. Assume $X_1 \cap V_B=\emptyset$ (the case of $X_2 \cap V_B=\emptyset$ is similar). If $C^+ \cap C'=\emptyset$ or $C^- \cap C'=\emptyset$ holds, then X_1 becomes a connected convergent set or a connected divergent set. By Lemma 2.3, $X_1 \cap V_B\neq\emptyset$ holds, but this contradicts the assumption. Therefore we have $C^+ \cap C'\neq\emptyset$ or $C^- \cap C'\neq\emptyset$. Then take an arc a_j $a(u_j,v_j) \in C^- \cap C'(=A(X_2))$ and a commodity $k' \in K(a_j)$ going out of $v_j \in X_1$, as shown in

Figure 6.20. By Lemma 6.9, $k' \notin K(a_i)$ holds for any $a_i \in C^+ \cap C'(=A(X_1))$. Then $t^{k'} \in V_B \subseteq X_2$ implies $\Pi^{k'}(v_j,t^{k'})=\emptyset$. By $s^{k'} \in REACH(x^*)$ and Lemma 6.11 (a), commodity k' must satisfy $t^{k'} \in Z^-$. There exists a node $z^* \in Z^-$ with $\Pi^{k'}(v_j,z^*)\neq\emptyset$ by Lemma 6.11 (b), and $z^* \in X_1$ holds by the reachability of $\Pi^{k'}$. As all convergent nodes are contained in V_B, node z^* is not convergent. By the acyclicity of G, we have a convergent node $w^* \in V_B \subseteq X_2$ with $\Pi(z^*,w^*)\neq\emptyset$. As cut C' separates z^* and w^*, C' and $\pi \in \Pi(z^*,w^*)$ have a common arc (say $a_h=a(u_h,v_h)$). By $z^* \in Z^-$, $z^* \notin REACH(x^*)$ holds, i.e., $u_h \notin REACH(x^*)$ and $TK(a_h)=\emptyset$ hold by Lemma 6.1. Since $r<q$, i.e., $C' \cap \{a_1,a_2,\ldots,a_r\}=\emptyset$, we obtain $h \geq r+1$. This, however, contradicts the property that $TK(a_i)\neq\emptyset$ for all $a_i \in C'$ $(i \geq r+1)$. □

Lemma 6.13 For $C^*=C^+ \cup C^-$ obtained by FIND-CUT,

(a) $TK(a_i) \subseteq X_1$ for $a_i \in C^-$,

(b) $TK(a_i) \subseteq X_2$ for $a_i \in C^+$.

Proof. We denote C^* by $\{a_1,a_2,\ldots,a_n\}$. By Lemma 6.8, the initial node u_n of arc a_n satisfies $u_n \in REACH(x^*)$. By Lemma 6.12, a_n is contained in W_B. Then, considering the partition of the boundary defined by C^*, $VB[\tilde{u},z'') \subseteq X_2$ and $VB(w'',\tilde{v}] \subseteq X_1$ hold, where $a_1=b_1=a(\tilde{u},\tilde{v}) \in A(V-X^-)$.

(a) Assume that there is some sink $t^{k'}$ with $t^{k'} \in TK(a_i) \cap X_2$ for some $a_i=a(u_i,v_i) \in C^-$. By $a_i \in C^-$ and Lemma 6.8, we have $t^{k'} \in TK(a_i) \subseteq VB(v_R(u_i),\tilde{v}] \subseteq VB(z'',\tilde{v}]$. Since no $a_j \in C^+$ satisfies $t^{k'} \in TK(a_j)$ by Lemma 6.9, $\Pi^{k'}(v_i,t^{k'})=\emptyset$ holds. Then $t^{k'} \in Z^-$ must hold by Lemma 6.11 (a). This means $t^{k'} \in VB[\tilde{u},z'')$ by the assumption $t^{k'} \in X_2$. This, however, contradicts $t^{k'} \in VB(z'',\tilde{v}]$.

(b) Assume that there is some sink $t^{k'}$ with $t^{k'} \in TK(a_i) \cap X_1$ for some $a_i=a(u_i,v_i) \in C^+$. By $a_i \in C^+$ and Lemma 6.8, we have $t^{k'} \in TK(a_i) \subseteq VB[\tilde{u},v_L(u_i)) \subseteq VB[\tilde{u},w'')$. Since no $a_j \in C^-$ satisfies $t^{k'} \in TK(a_j)$ by Lemma 6.9, we get $\Pi^{k'}(v_i,t^{k'})=\emptyset$. Then $t^{k'} \in Z^-$ must hold by Lemma 6.11 (a). This meams $t^{k'} \in VB(w'',\tilde{v}]$ by the assumption $t^{k'} \in X_1$. This, however contradicts $t^{k'} \in VB[\tilde{u},w'')$. □

Lemma 6.14 Let C^* be the set of arcs obtained by FIND-CUT. Then C^* is a simple cut, but not unilateral. Furthermore, C^* is a commodity disjoint cut and satisfies $|C^* \cap A_B|=2$.

Proof. By Lemma 6.12, C^* is a simple cut and $C^* \cap A_B=\{a_1,a_n\}$ holds. C^* is not unilateral, since $a_1 \in C^-$ and $a_{r+1} \in C^+$ hold. By Lemmas 6.8 and 6.9, flows are assigned to all the arcs in $A(X_1)=C^+$, and $TK(a_i) \cap TK(a_j)=\emptyset$ holds for $a_i \in C^-$ and $a_j \in C^+$. By Lemma 6.13 (b),

$TK(a) \subseteq V - X_1$ for all $a \in A(X_1)$. As a result of this, cut C^* satisfies (6.1)-(6.2), and then is commodity disjoint. □

6.5 MAX-FLOW MIN-CUT THEOREM FOR CB AND CS NETWORKS

Based on the results in the previous sections, we shall prove Theorems 6.1 and 6.2 stated in Section 6.2, i.e., the max-flow min-cut theorem for CB and CS networks.

Lemma 6.15 An infeasible CB network N" satsfying Assumption B has the following simple cut C^*.

(i) C^* is not unilateral, and $|C^* \cap A_B| = 2$.

(ii) C^* does not satisfy the cut-condition (2.1).

Proof. When ASSIGN halts in N", we construct (N,f) according to Definition 6.1. In N, FIND-CUT finds a simple cut C^* which is not unilateral. C^* partitions V into X_1 and $X_2(=V-X_1)$ such that $A(X_1)=C^+$ and $A(X_2)=C^-$. By Lemma 6.13, C^* is a commodity disjoint cut and satisfies (i) of the lemma's statement. We now show that C^* does not satisfy the cut-condition (2.1). Since all arcs in $A(X_1)$ are saturated,

$$c(X_1) = \sum_{k \in K(a)} \sum_{a \in A(X_1)} f(a,k)$$

holds. As $TK(a) \subseteq V-X_1$ for $a \in A(X_1)$ by Lemma 6.12, clearly

$$g(X_1) \geq \sum_{k \in K(a)} \sum_{a \in A(X_1)} f(a,k)$$

holds. Here $a(x^*,v^*)$ is located to the left of $a(x^*,z_1)$ by the definition of $a_{r+1}=a(x^*,z_1)$, and therefore C^* does not contain $a(x^*,v^*)$ by Lemma 6.10. That is, $v^* \in X_1$ holds. Since $f(a(x^*,v^*))>0$ holds for commodity k^* that satisfies the termination condition (T-4), this flow does not outgo from node $v^* \in Z^-$. Therefore, by considering $t^{k^*} \in X_2$, $g(X_1)$ can be estimated as follows.

$$g(X_1) \geq \sum_{k \in K(a)} \sum_{a \in A(X_1)} f(a,k) + f(a(x^*,v^*),k^*).$$

Here, the r-value of X_1 becomes

$$r(X_1)=c(X_1)-g(X_1)= \sum_{k \in K(a)} \sum_{a \in A(X_1)} f(a,k) - g(X_1)$$
$$\leq -f(a(x^*,v^*),k^*)<0.$$

That is, C^* does not satisfy the cut-condition. By Lemma 2.6, C^* also satisfies (i) in N". Each commodity whose source and sink are separated by C^* in N remains still in N". Therefore, C^* does not satisfy the cut-condition (2.1) in N". □

Theorem 6.1 follows immediately from Lemma 6.15. Now we prove Theorem 6.2 for CS networks.

Lemma 6.16 If a CS network N' does not satisfy Assumption B, then N' has a unilateral simple cut not satisfying the cut-condition (2.1).

Proof. Obvious from the argument prior to Assumption B in Section 2.4. □

Lemma 6.17 An infeasible CS network N' satisfying Assumption B has a simple cut not satisfying the cut-condition (2.1).

Proof. In Chapter 4, we presented algorithm MATE1 that reduces a CS network to a CB network. Let N" be the CB network reduced from N' according to MATE1, and let K^* denote the set of commodities created by MATE1 in N". If N" does not satisfy Assumption B, N" has a unilateral simple cut not satisfying the cut-condition as described in Section 2.4. Otherwise, if N" satisfies Assumption B, then, by Lemma 6.15, N" has a non-unilateral simple cut $C^*=A(X_1) \cup A(X_2)$ not satisfying the cut-condition (2.1). Here, by Lemma 2.8, we have $r(X_1)=r(X_2)<0$. If $K(X_1;X_2) \cap K^*=\emptyset$ holds, then C^* is also a simple cut not satisfying the cut-condition since the value of $r(X_1)$ does not change in N'. Since case $K(X_2;X_1) \cap K^*=\emptyset$ can be treated similarly, assume that $K(X_1;X_2) \cap K^* \neq \emptyset$ and $K(X_2;X_1) \cap K^* \neq \emptyset$,

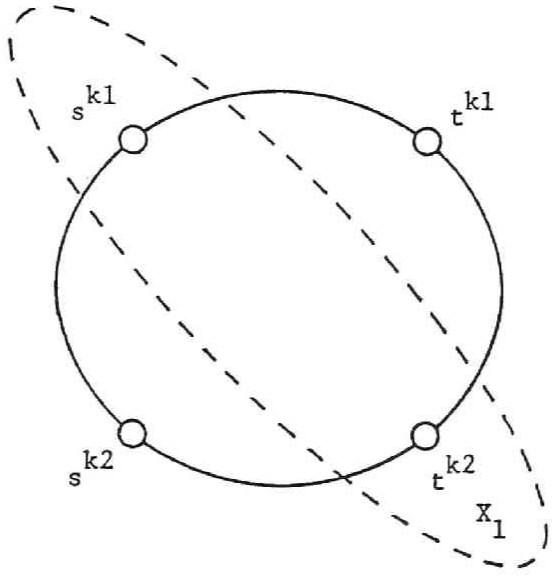


Fig. 6.21 Proof of Lemma 6.17.

and take $k1 \in K(X_1;X_2) \cap K^*$ and $k2 \in K(X_2;X_1) \cap K^*$. In a CS network, the boundary line can be divided into two parts such that all dummy sources are located in one part while all dummy sinks are located in the other part. Source-sink pair of each commodity in K^* was a neighbouring pair before

its commodity is created by MATE1. This implies that a path between source and sink of a new commdity $k1 \in K^*$ does not cross a path between source-sink pair of other new commdity $k2 \in K^*$. That is, s^{k1}, t^{k1}, s^{k2} and t^{k2} must be located in the boundary as shown in Figure 6.21. Then we have $|C^* \cap A_B| \geq 4$ from $s^{k1}, t^{k2} \in X_1$. This, however contradicts $|C^* \cap A_B| = 2$ of Lemma 6.15 (i). Therefore we obtain $K(X_1;X_2) \cap K^* = \emptyset$ and $K(X_2;X_1) \cap K^* = \emptyset$. □

Lemmas 6.16 and 6.17 give Theorem 6.2.

6.6 CONCLUSION

In this chapter, we showed that CB and CS are classes of multicommodity flow problems, for which the max-flow min-cut theorem holds. It should be noted that algorithm ASSIGN for CB does not employ the max-flow min-cut property. Based on the max-flow min-cut property, the feasibility of CB can be tested by computing the minimum cut immediately. In the next chapter, we construct this type of algorithm for classes CB and CS.

CHAPTER 7

ALGORITHM FTEST FOR TESTING FEASIBILITY OF A CB OR CS NETWORK

7.1 INTRODUCTION

It was shown in the previous chapter that the max-flow min-cut theorem holds for classes CB and CS. Based on the max-flow min-cut property, we shall present in this chapter an $O(S(|V|)+|V_B|T(|V|))$ time algorithm for testing feasibility of the multicommodity flow problem for classes CB and CS, where $|V|$ is the number of nodes, $|V_B|$ is the number of nodes in the boundary, $S(|V|)$ is the time required for preprocessing and $T(|V|)$ is the time to compute a shortest path tree. The best bounds currently known are $S(|V|)=O(1)$, $T(|V|)=O(|V|\sqrt{\log|V|})$ and $S(|V|)=O(|V|\log|V|)$, $T(|V|)=O(|V|\log^*|V|)$, which respectively give $O(|V_B||V|\sqrt{\log|V|})$ and $O(|V|\log|V|+|V_B||V|\log^*|V|)$ time bounds for our problem. These are better than the bound $O(K|V|)$ obtained in Chapter 3, when the number of commodities K is large [NAGA 88b].

7.2 ASSUMPTION D AND SOME LEMMAS

In the subsequent discussion, we consider CB and CS networks satisfying the following assumption.

Assumption D: $S \cup T \subseteq V_B$. □

Under this assumption, $g(X)=0$ clearly holds for an arbitrary set $X \subseteq V-V_B$. That is, such set of nodes X satisfies cut-condition (2.1) ($r(X)=c(X)\geq 0$). Therefore, to examine condition (2.1), we concentrate on X with $X \cap V_B \neq \emptyset$. Furthermore, we make use of the following property.

Lemma 7.1 In a connected graph G, $r(X)\geq 0$ holds for every $X \subseteq V$, if and only if $r(X)\geq 0$ holds for those $X \subseteq V$ such that X and V-X are connected, respectively.

Proof. As necessity is obvious, we show sufficiency. First, we show that $r(Y)\geq 0$ and $r(Z)\geq 0$ imply $r(Y \cup Z)\geq 0$ for any $Y,Z \subseteq V$ with $Y \cap Z=\emptyset$ and $A(Y;Z)=A(Z;Y)=\emptyset$. Now

$$\begin{aligned} r(Y \cup Z) &= c(Y \cup Z) - g(Y \cup Z) \\ &= c(Y) + c(Z) - \{g(Y)+g(Z)-g(Y;Z)-g(Z;Y)\} \\ &= r(Y) + r(Z) + g(Y;Z) + g(Z;Y) \\ &\geq r(Y) + r(Z) \geq 0. \end{aligned}$$

If a set $X \subseteq V$ is composed of some number of connected sets, repeat the above argument. It shows that if $r(X)\geq 0$ holds

for any connected set X then $r(X) \geq 0$ for every $X \subseteq V$.

Now we assume that X is connected, but V-X is not connected. Then V-X may be divided into disjoint connected components. For convenience, we assume that the number of such components are two (say sets Y and Z). That is, $X \cap Y = X \cap Z = Y \cap Z = \emptyset$, $X \cup Y \cup Z = V$ and $A(Y;Z) = A(Z;Y) = \emptyset$. For these, we obtain $A(X;Y) \cup A(Y;X) \subseteq A(X) \cup A(V-X)$ and $\{A(Z;X) \cup A(X;Z)\} \cap \{A(X;Y) \cup A(Y;X)\} = \emptyset$. This means that V-Y is connected, because $A(Z;X) \cup A(X;Z) \neq \emptyset$ holds since G is connected. Similarly we can show that V-Z is connected. Since these sets satisfy the condition of the lemma, we can assume $r(V-Y) \geq 0$ and $r(V-Z) \geq 0$. Then we have

$$\begin{aligned} r(X) &= r(V-(Y \cup Z)) = c(V-(Y \cup Z)) - g(V-(Y \cup Z)) \\ &= c(V-Y) + c(V-Z) - \{g(V-Y)+g(V-Z)-g(Y;Z)-g(Z;Y)\} \\ &= r(V-Y) + r(V-Z) + g(Y;Z) + g(Z;Y) \\ &\geq r(V-Y) + r(V-Z) \geq 0. \end{aligned}$$

Therefore, we obtain that if $r(X) \geq 0$ for $X \subseteq V$ such that X and V-X are both connected, then $r(X) \geq 0$ holds for any $X \subseteq V$. □

By combining this result and Assumption D, we conclude the following. To test feasibility, it is sufficient to examine (2.1) only for connected sets $X \subseteq V$ with $X \cap V_B =$

VB[x,y) for some $x,y \in V_B$. Here, we consider the case in which $x,y \in V_B$ are fixed. Then, for any X with $X \cap V_B =$ VB[x,y), we have

$$r(X) = c(X) - g(X) = c(X) - g(VB[x,y))$$ (see Figure 7.1).

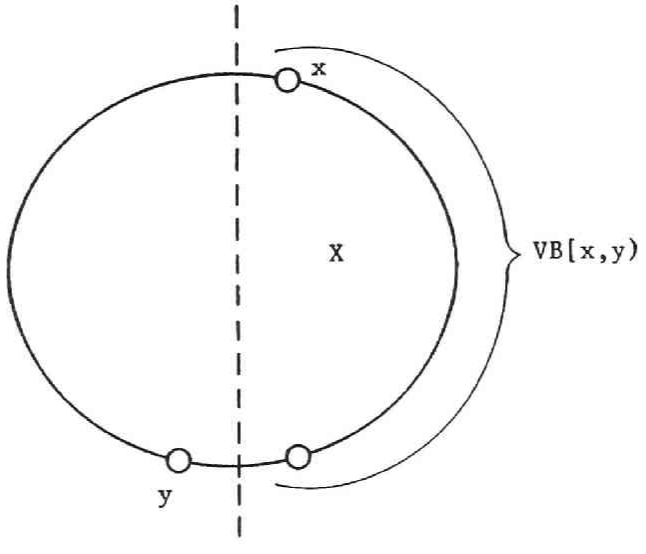


Fig. 7.1 Illustration of VB[x,y).

Therefore, to test $r(X) \geq 0$, we consider that X minimizes $c(X)$. Let $c^{o}[x,y)$ denote the minimum value of $c(X)$, then we obtain the following lemma.

Lemma 7.2 A CB (or CS) network is feasible, if and only if

$$c^{o}[x,y)-g(VB[x,y)) \geq 0 \quad \text{for all } x,y \in V_B,\ x \neq y \qquad (7.1)$$

holds. □

7.3 USE OF A DUAL GRAPH

In this section, we consider the use of a dual graph in order to test condition (7.1) efficiently.

Definition 7.1 For a CB or CS network $N=(G,P,g,c)$ satifying Assumption D, we define directed dual graphs G^*, H^* and directed networks N^* as follows. Each node v_i^* in the dual graph corresponds to a window W_i in G (in particular, node v_B^* corresponds to the outer face of G). For each directed arc a_k in G, a directed arc a_k^+ is introduced between nodes v_i^* and v_j^* corresponding to the windows W_i and W_j which contain a_k commonly. Here, direction of a_k^+ is defined by the orientation when we turn a_k 90 degrees in the clockwise

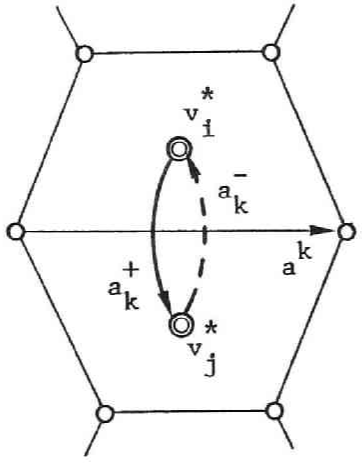


Fig. 7.2 Explanation of Definition 7.1.

manner (see Figure 7.2). Let V^* and A^+ denote the resulting set of nodes and set of arcs, respectively. Further, let a_k^- be the arc obtained by reversing the orientaion of a_k^+, and let A^- be the set of arcs a_k^-. We denote dual graphs $(V^*, A^+ \cup A^-)$ by G^*. Furthermore, let H^* denote the resulting graph obtained by removing v_B^* and all arcs incident to v_B^* from G^*. By introducing the length of arcs $d: A^+ \cup A^- \rightarrow \mathbf{R}^+$, we define directed dual networks $N^*=(H^*,d)$, where d is defined by

$$d(a_i^+)=c(a_i) \quad \text{for } a_i^+ \in A^+,$$

$$d(a_i^-)=0 \quad \text{for } a_i^- \in A^-. \quad \square$$

Assume that we know the information about the incidence relation of arcs, which is necessary to embed a planar graph into the plane. Then, since $O(|V^*|)=O(|A^+|)=O(|A^-|)=O(|A|)=O(|V|)$ holds in a planar graph, we can find all windows in G and construct G^* according to Definition 7.1 in $O(|V|)$ time.

We take a simple directed cycle C in G^* and consider the set of arcs in G corresponding to the arcs in C. The obtained set is a simple cut in G, and from definition of d, the length of the directed cycle is equal to the sum of capacities of arcs in the simple cut. Since any cut

necessary for examining (7.1) contains an arc in the boundary, we shall concentrate on, in particular, directed cycles containing v_B^* among all simple directed cycles in G^*. As shown in Figure 7.3, let $x_1,x_2,\ldots,x_m$ ($m=|V_B|$) denote the nodes which appear in this order when we go around the boundary in the clockwise manner. Further, we denote the directed arc between x_i and x_{i+1} by a_i with convention $m+1=1$. Let v_i, $i=1,2,\ldots,m$ denote the node adjecent to v_B^* in G^* via $a_i^+ \in A^+$, $i=1,2,\ldots,m$, where a_i^+, $i=1,2,\ldots,m$ in G^* correspond to the above $a_i \in A$, $i=1,2,\ldots,m$. Let V_B^* be the set of v_i^*, $i=1,2,\ldots,m$ ($v_i^*=v_j^*$ is possible for $i\neq j$, but similar argument also applies).

Lemma 7.3 Let $d^o(v_j^*,v_i^*)$ be the length of the shortest directed path from v_j^* to v_i^* in N^*. Then

$$c^o[x_{i+1},x_{j+1}) = d^o(v_j^*,v_i^*) + d(a(v_B^*,v_j^*)) + d(a(v_i^*,v_B^*))$$

holds, where $m+1=1$ is assumed for convenience.

Proof. In (G^*,d), consider a directed cycle C such that the sum of arc lengths in C is 0. Then all arcs in C are in A^-. Since C is a directed cycle in graph (V^*,A^-), this means that C corresponds to a cut A(X) between a divergent set X and a convergent set V-X in G. Therefore $A(X) \cap A_B \neq \emptyset$ by Lemma 2.4, that is, any directed cycle in

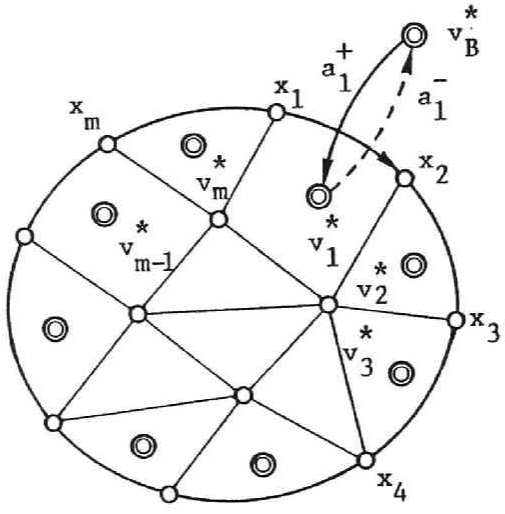


Fig. 7.3 Definition of $v_1^*, v_2^*, \ldots, v_m^*$.

graph (G^*,d) with length 0 must contain v_B^*. Hence, if a directed cycle does not contain v_B^*, then this cycle has a positive sum of lengths. Thus, this implies that the shortest path (with $d^o(v_j^*,v_i^*)$ in length) described in the lemma is simple. Therefore, the shortest path from v_j^* to v_i^* and two arcs $a(v_B^*,v_j^*)$, $a(v_i^*,v_B^*)$ give a directed cycle in G^*, and there is a simple cut in N with its cost $c^o[x_{i+1},v_{j+1})$

corresponding to the directed cycle. By Definition 7.1, C^o contains arcs $a(v_B^*,v_j^*)$ and $a(v_i^*,v_B^*)$. Thus, C^o contains subgraph $v_B^* \rightarrow v_j^* \rightarrow v_i^* \rightarrow v_B^*$. Therefore, by the minimumity of $c^o[x_{i+1},x_{j+1})$, C^o and C' have the same cost value and we have the lemma's statement. □

Lemma 7.3 asserts that we can compute $c^o[x,y)$, by constructing the shortest path tree $ST(v_i^*)$ in N^* for each of i=1,2,...,m, where $ST(v_i^*)$ is the shortest path tree from root v_i^* to all other v_j^*. As N^* is strongly connected by the definition, each $ST(v_i^*)$ contains all nodes in N^*, where a graph G=(V,A) is strongly connected if $\Pi(x,y)\neq\emptyset$ holds for any pair $x,y \in V$, $x\neq y$. Let $T(|V|)$ and $S(|V|)$ denote the time required to compute a shortest path tree and the preprocessing time for it, repectively. We can obtain all shortest path trees $ST(v_i^*)$, $i=1,2,\ldots,|V_B|$, in $O(S(|V|)+|V_B|T(|V|))$ time.

7.4 COMPUTATION OF g(VB[x,y))

To obtain all $g(VB[x,y))$, we first compute $g(VB[x_1,x_i))$, $i=2,3,\ldots,m$ for $x_1 \in V_B$ in $O(K+|V_B|)$ time as follows. Introduce 0-1 variables $\delta(k)$ for commodities $k=1,2,\ldots,K$, and let

$$K_s(x) \stackrel{\Delta}{=} \{k \mid s^k=x\}, \quad K_t(x) \stackrel{\Delta}{=} \{k \mid t^k=x\} \quad \text{for each } x \in V_B.$$

Procedure SETG(x_1)

0. $\delta(k):=1$ for $k=1,2,\ldots,K$;

   $g(VB[x_1,x_1)):=0$; $i:=1$.

1. $$g(VB[x_1,x_{i+1})):=g(VB[x_1,x_i)) + \sum_{k \in K_s(x_i)} \delta(k)g^k - \sum_{k \in K_t(x_i)} (1-\delta(k))g^k;$$

   $\delta(k):=1-\delta(k)$ for all $k \in K_s(x_i) \cup K_t(x_i)$.

2. If $i=m-1$ then halt the procedure. Otherwise let $i:=i+1$ and return to Step 1. □

This procedure is executed from x_1 to x_m along the boundary in the clockwise manner. We first consider the case in which source and sink of some commodity k are located in the order of $x_1,\ldots, x_p=s^k,\ldots,x_q=t^k,\ldots,x_m$. Since $\delta(k)$ is set to 1 in Step 0 and changes only when x_p or x_q is scanned, the value of g^k is added only to

$g(VB[x_1,x_i])$, $i=p+1,\ldots,q$. In case the source and sink of some commodity k are located in the order of $x_1,\ldots,x_p=t^k,\ldots,x_q=s^k,\ldots,x_m$, the value of $\delta(k)=1$ changes to 0 (1) when x_p (x_q) is scanned, and the value of g^k is not added to any $g(VB[x_1,x_i])$, $i=2,3,\ldots,q$. Therefore, the procedure SET(x_1) computes $g(VB[x_1,x_i])$, $i=2,3,\ldots,q$, correctly. By storing $K_s(x_i)$ and $K_t(x_i)$ for each $x_i \in V_B$ in advance and referring the data as required, the total time of procedure SETG(x_1) is $O(K+|V_B|)$.

If $g(VB[x_1,x_i])$, $i=2,3,\ldots,m$ for $x_1 \in V_B$ is known, we can obtain $g(VB[x_2,x_i])$, $i=3,4,\ldots,m,1$ for $x_2 \in V_B$ as follows.

Procedure UPDATEG(x_2)

0. $g(VB[x_2,x_1]) := \sum_{k \in K_t(x_1)} g^k$;

$\Delta_2 := - \sum_{k \in K_s(x_1)} g^k$;

$i:=3$.

1. $\Delta_i := \Delta_{i-1} + \sum_{k' \in K_t(x_1) \cap K_s(x_{i-1})} g^{k'} + \sum_{k \in K_s(x_1) \cap K_t(x_{i-1})} g^k$;

$g(VB[x_2,x_i]) := g(VB[x_1,x_i]) + \Delta_i$.

2. If $i=m$ then halt the procedure. Otherwise let $i:=i+1$ and return to Step 1. □

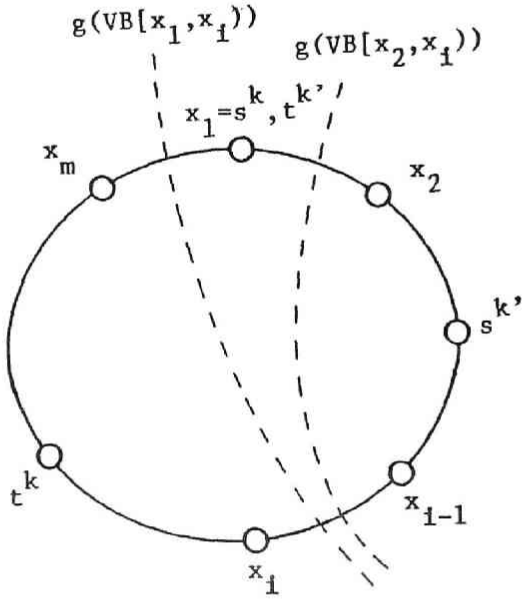


Fig. 7.4 Explanation of procedure UPDATEG(x_2).

For i=3,4,...,m,1, we clearly have

$$g(VB[x_2,x_i))=g(VB[x_1,x_i)) + \sum_{k'\in K_t(x_1)\cap\{K_s(x_2)\ \ldots\ K_s(x_{i-1})\}} g^{k'} + \sum_{k\in K_s(x_1)\cap\{K_t(x_i)\ \ldots\ K_t(x_m)\}} g^{k}.$$

See Figure 7.4. Since $K_s(x_1)\cap\{K_t(x_i)\cup\ldots\cup K_t(x_m)\}$ can be rewritten as $K_s(x_1)-K_s(x_1)\cap\{K_t(x_i)\cup\ldots\cup K_t(x_m)\}$, each increment Δ_i of $g(VB[x_2,x_i))-g(VB[x_1,x_i))$ satisfies the relation in Steps 0 and 1. This shows the validity of procedure UPDATEG. Since we may find x_i with $k\in K_t(x_i)$ $(K_s(x_i))$ in O(1) time for each $k\in K_s(x_1)$ $(K_t(x_1))$ by using appropriate data structure, the required time of UPDATEG is bounded from above by $O(|K_s(x_1)|+|K_t(x_1)|+|V_B|)$.

By repeating this procedure UPDATEG for $x_3,x_4,\ldots,x_m$, therefore we can obtain all $g(VB[x,y))$ within $O(K+|K_s(x_1)|+\ldots+|K_s(x_{m-1})|+|K_t(x_1)|+\ldots+|K_t(x_{m-1})|+|V_B|^2)$ $=O(K+|V_B|^2)$ time.

7.5 FEASIBILITY TESTING

By the following procedure FTEST, which is based on the procedures in the previous section, we can test feasibilities of a CB network or a CS network $N=(G,P,g,c)$.

Procedure FTEST

Input: A CB or CS network $N=(G,P,g,c)$ satisfying Assumption D.

Output: Feasibility of N.

0. Construct dual networks $N^*=(H^*,d)$ and according to Definition 7.1.

1. Obtain the shortest path trees $ST(v_i^*)$ with root v_i^*, $i=1,2,\ldots,m$ in N^*.

2. Compute all $g(VB[x,y))$, $x,y \in V_B$ $(x \neq y)$ by SETG and UPDATEG.

3. For each pair $x_{i+1},x_{j+1} \in V_B$ in N, compute $c^o[x_{i+1},x_{j+1})$ by Lemma 7.3 and $ST(v_i^*)$, and test

$$c^o[x_{i+1},x_{j+1})-g(VB[x_{i+1},x_{j+1}))\geq 0$$

If this holds for all $[x_{i+1},x_{j+1})$, then output "feasible" and halt. Otherwise, output "infeasible" and halt. □

The required time of the above computation is $O(|V|)+O(S(|V|)+|V_B|T(|V|))+O(K+|V_B|^2)$ as obvious from the results

in Sections 7.3-7.4. Here, if we do not distinguish the commodities with the same source-sink pair, we have

$$K \leq |V_B|^2.$$

In addition, suppose $T(|V|) \geq O(|V|)$. Then the entire time then becomes $O(S(|V|)+ |V_B|T(|V|))$.

Theorem 7.1 Feasibility of a CB (CS) network N can be tested in $O(S(|V|)+|V_B|T(|V|))$ time, where $|V|$ is the number of nodes, $|V_B|$ is the number of nodes in the boundary, $T(|V|)$ and $S(|V|)$ are the time required to compute a shortest path tree and the preprocessing time for it. □

As N^* is a planar graph with nonnegative arc lengths, Frederickson's two algorithms [FRED 83] have running times $S(|V|)=O(1)$, $T(|V|)=O(|V|\sqrt{\log|V|})$ and $S(|V|)=O(|V|\log|V|)$, $T(|V|)= O(|V|\log^*|V|)$, where $\log^* n$ is the minimum integer p satisfying $\log^{(p)} n \leq 1$, assuming that $\log^{(i)} n$ is defined as $\log^{(0)} n=\log n$ and $\log^{(i)} n=\log(\log^{(i-1)} n)$, $i \geq 1$. As described in Section 7.1, the time complexities obtained from these results of $S(|V|)$ and $T(|V|)$ become advantageous when the number of commodities K is large.

Finally, consider the scheduling network in Figure 4.2 and apply FTEST. Since $|V_B|=O(I+J)$ and $|V|=O(IJ)$. the

latter of Frederickson's gives the time complexity $O((I+J)IJ\log^{*}IJ)$ of FTEST. For large K, this new time bound becomes smaller if compared with the O(IJK) time bound of the algorithm developed exclusively for this sheduling problem in [IBAR 85], and the $O(K|V|+|V_B||V|)=O((K+I+J)IJ)$ time bound of the algorithms ASSIGN and MATE1 developed in Chapters 3 and 4 for CS.

7.6 CONCLUSION

We showed that the feasibility of a network in class CB or CS of the multicommodity flow problem in directed planar graphs can be tested efficiently by computing the shortest path trees in their dual graphs. This result is based on the max-flow min-cut property that holds for these classes. If the number of commodities K is large, the proposed method FTEST is more efficient than the methods based on ASSIGN and MATE1. It should be noted, however, that FTEST answers only whether the network is feasible or not, and does not output the values of feasible flow f(a,k) even if it is feasible. This point differs from the algorithm based on ASSIGN and MATE1.

CHAPTER 8

RELAXATION METHODS

FOR A DIRECTED NONLINEAR NETWORK

8.1 INTRODUCTION

As described in Chapter 1, the multicommodity flow problem can be formulated as a linear programming (LP) problem if its objective function is linear. Some graph theoretic algorithms are also developed in Chapters 3-7 of this thesis. If the objective function is not linear, these algorithms for the LP formulation can no longer be applied.

In case of nonlinear objective functions, Bertsekas [BERT 85,87] considers the dual problem for the minimum cost single commodity flow problem with strictly convex objective function, which is based on the dual formulation [ROCK 81]. As to the multicommodity flow problem, some methods have been developed for the case where the capacity restriction is imposed only on the total flow of commodities in each arc. (See, for example, the dual approach by Fukushima [FUKU 84] and the relaxation approaches by Gallager [GALL 77] and Stern [STER 77].) Fukushima's alogorithm [FUKU 84] obtains an optimal solution of the dual problem by repeatedly solving the shortest path

problems.

In this chapter, we propose relaxation methods for the multicommodity flow problem with a strictly convex objective function. Associated with each arc in the network are capacity constraints not only on the total flow of commodities but also on the flow of each commodity. Capacity constraints for individual commodities are often required, for example, in order to represent a selection of particular commodities in an arc of the network for multi-stage multi-item production scheduling problem [ZAHO 84]. The methods proposed in this chapter are extensions of the methods proposed by Bertsekas [BERT 85,87] for the single commodity case, and are efficient enough to practically solve problem instances of moderate sizes with several commodities. Typical computational results show that problem instances with up to 100 nodes, 1000 arcs and 7 commodities can be solved in about 60 seconds on a FACOM M-382 machine [NAGA 87c].

It is noted here that the previous formulations such as those considered in [FUKU 84, GALL 77. STER 77] are different from ours because they do not have capacity bounds on individual commodities, and their objective functions

contain as arguments only the total flows of commodities on arcs. In this sense, our formulation is more general. However, their formualtions are not special cases of ours because the deletion of the objective function of individual commodities (i.e., to assume that it is constant) violates the assumption of strict convexity, which is vital to the proposed methods.

8.2 DEFINITIONS

We consider the following network $N=(G,\underline{c},\bar{c})$.

$G=(V,A)$: a directed graph, where V is the set of nodes and A is the set of arcs. $(i,j) \in A$ denotes a directed arc from node i to node j.

$\underline{c}$: a vector of lower capacities. $\underline{c}_{ij}^{k}$ denotes the lower bound for commodity k in arc (i,j), and $\underline{c}_{ij}$ denotes the lower bound for the sum of all commodities in arc (i,j). (Elements of $\underline{c}$ are allowed to be $-\infty$)

$\bar{c}$: a vector of upper capacities. $\bar{c}_{ij}^{k}$ denotes the upper bound for commodity k in arc (i,j), and $\bar{c}_{ij}$ denotes the upper bound for the sum of all commodities in arc (i,j). (Elements of $\bar{c}$ are allowed to be $+\infty$)

Let K be the set of commodities, and let f_{ij}^{k} denote the flow of commodity $k \in K$ in arc (i,j). The multicommodity flow circulation problem we consider is stated as follows.

$$\hat{P}: \text{ minimize } \sum_{(i,j)\in A} \sum_{k\in K} \hat{F}_{ij}^{k}(f_{ij}^{k}) + \sum_{(i,j)\in A} \hat{G}_{ij}(s_{ij})$$

subject to

$$\sum_{(i,m)\in A} f_{im}^{k} - \sum_{(m,i)\in A} f_{mi}^{k} = 0 \qquad \text{for } k \in K,\ i \in V, \quad (8.1)$$

$$s_{ij} = \sum_{k\in K} f_{ij}^{k} \qquad \text{for } (i,j) \in A, \quad (8.2)$$

$$\underline{c}_{ij}^{k} \le f_{ij}^{k} \le \bar{c}_{ij}^{k} \qquad \text{for } k \in K,\ (i,j) \in A, \quad (8.3)$$

$$\underline{c}_{ij} \le s_{ij} \le \bar{c}_{ij} \qquad \text{for } (i,j) \in A, \quad (8.4)$$

where $\hat{F}_{ij}^{k}: R \to (-\infty, +\infty)$ and $\hat{G}_{ij}: R \to (-\infty, +\infty)$ are (everywhere finite) strictly convex functions. Further, we assume

$$\lim_{x \to -\infty} \hat{F}_{ij}^{k}(x)/x = -\infty \text{ if } \underline{c}_{ij}^{k} = -\infty,$$

$$\lim_{x \to -\infty} \hat{G}_{ij}(x)/x = -\infty \text{ if } \underline{c}_{ij} = -\infty,$$

$$\lim_{x \to +\infty} \hat{F}_{ij}^{k}(x)/x = +\infty \text{ if } \bar{c}_{ij}^{k} = +\infty,$$

$$\lim_{x \to +\infty} \hat{G}_{ij}(x)/x = +\infty \text{ if } \bar{c}_{ij} = +\infty. \quad (8.5)$$

Strict convexity and assumption (8.5) on $\hat{F}_{ij}^{k}$ and $\hat{G}_{ij}$ are necessary to guarantee that their conjugate functions are finite everywhere and continuously differentiable. Constraint (8.1) is the flow conservation of commodity k at node i. Note that the ordinary multicommodity flow problem, in which some nodes are specified as sources or sinks, is reducible to this formulation $\hat{P}$ by introducing return arcs with appropriate lower capacities from sinks to sources.

Let f denote the $|K||A|+|A|$ dimensional vector with components f_{ij}^{k} ($k \in K$, $(i,j) \in A$) and s_{ij} ($(i,j) \in A$). Problem $\hat{P}$ may then be rewritten as follows.

$$\text{P: minimize } \phi(f) \overset{\Delta}{=} \sum_{(i,j)\in A} \sum_{k\in K} F_{ij}^k(f_{ij}^k) + \sum_{(i,j)\in A} G_{ij}(s_{ij}) \tag{8.6}$$

subject to

$$\sum_{(i,m)\in A} f_{im}^k - \sum_{(m,i)\in A} f_{mi}^k = 0 \quad \text{for } k\in K,\ i\in V, \tag{8.7}$$

$$s_{ij} = \sum_{k\in K} f_{ij}^k \quad \text{for } (i,j)\in A, \tag{8.8}$$

where $F_{ij}^k: R \to (-\infty, +\infty]$ and $G_{ij}: R \to (-\infty, +\infty]$ are defined by

$$F_{ij}^k(x) = \begin{cases} \hat{F}_{ij}^k(x), & x \in [\underline{c}_{ij}^k, \tilde{c}_{ij}^k] \\ +\infty, & x \notin [\underline{c}_{ij}^k, \tilde{c}_{ij}^k] \end{cases}$$

$$G_{ij}(x) = \begin{cases} \hat{G}_{ij}(x), & x \in [\underline{c}_{ij}, \tilde{c}_{ij}] \\ +\infty, & x \notin [\underline{c}_{ij}, \tilde{c}_{ij}]. \end{cases}$$

Let p denote the $|K||V|+|A|$ dimensional vector with components p_i^k ($k\in K$, $i\in V$) and p_{ij} ($(i,j)\in A$). We then derive the dual D of problem P as follows. Consider the Lagrangean of P and its inf:

$$L_0(f,p) = \phi(f) + \sum_{(i,j)\in A} \sum_{k\in K} p_i^k \Big(- \sum_{(i,m)\in A} f_{im}^k + \sum_{(m,i)\in A} f_{mi}^k\Big) + \sum_{(i,j)\in A} p_{ij}\Big(-s_{ij} + \sum_{k\in K} f_{ij}^k\Big),$$

$$\inf_{f} L_0(f,p)$$

$$= \sum_{(i,j)\in A} \sum_{k\in K} \inf_{f_{ij}^k} \{F_{ij}^k(f_{ij}^k) + (-p_i^k+p_j^k+p_{ij})f_{ij}^k\}$$

$$+ \sum_{(i,j)\in A} \inf_{s_{ij}} \{G_{ij}(s_{ij})-p_{ij}s_{ij}\}$$

$$= - \sum_{(i,j)\in A} \sum_{k\in K} F_{ij}^{*k}(p_i^k-p_j^k-p_{ij}) - \sum_{(i,j)\in A} \{G^*_{ij}(p_{ij})\}.$$

Here F_{ij}^{*k} and G_{ij}^* are the functions conjugate to F_{ij}^k and G_{ij}, respectively. By the theory of nonlinear programming, therefore, D can be written as follows.

$$\text{D: minimize} \quad \sum_{(i,j)\in A} \sum_{k\in K} F_{ij}^{*k}(t_{ij}^k) + \sum_{(i,j)\in A} G_{ij}^*(p_{ij}) \quad (8.9)$$

subject to

$$t_{ij}^k = p_i^k - p_j^k - p_{ij} \qquad \text{for } k\in K,\ (i,j)\in A. \quad (8.10)$$

One may consider the variable p_i^k as representing the potential of commodity k at node i, as in the case of the single commodity flow problem [ROCK 84]. The variable p_{ij} correponding to constraint (8.2) is unique to the multicommodity flow problem. Here we shall call variables p_{ij} the potential of arc (i,j). We also call variables t_{ij}^k satisfying (8.10) the tension of arc (i,j).

Substituting (8.10) into the objective function, problem D becomes an unconstrained minimization problem with

variables p_i^k and p_{ij}:

$$\text{minimize } \xi(p) \triangleq \sum_{(i,j) \in A} \sum_{k \in K} F_{ij}^{*k}(p_i^k - p_j^k - p_{ij}) + \sum_{(i,j) \in A} G_{ij}^{*}(p_{ij}). \tag{8.11}$$

It is known [ROCK 70; Corollary 13.3.1, Corollary 25.5.1 and Theorem 26.3] that assumption (8.5) implies that F_{ij}^{*k} and G_{ij}^{*} are continuously differentiable convex functions which take finite values everywhere. In the convergence proof of the alogorithms to be presented in Section 8.4, we shall also assume that the first derivatives of the functions are Lipschitz continuous.

8.3 OPTIMALITY CONDITIONS

Let (t,p) denote the vector of all variables t_{ij}^k, p_i^k and p_{ij} of D. By the duality theory of nonlinear programming, a solution f of P and a solution (t,p) of D are both optimal if

$$t_{ij}^k = p_i^k - p_j^k - p_{ij} \qquad \text{for } k \in K,\ (i,j) \in A, \quad (8.12)$$

$$dF_{ij}^{*k}(t_{ij}^k)/dt_{ij}^k - f_{ij}^k = 0 \qquad \text{for } k \in K,\ (i,j) \in A, \quad (8.13)$$

$$\sum_{(i,m) \in A} f_{im}^k - \sum_{(m,i) \in A} f_{mi}^k = 0 \qquad \text{for } k \in K,\ i \in V, \quad (8.14)$$

$$dG_{ij}^*(p_{ij})/dp_{ij} - \sum_{k \in K} f_{ij}^k = 0 \qquad \text{for } (i,j) \in A, \quad (8.15)$$

hold. In particular, conditions (8.12) and (8.13) imply that t of D and f of P can both be determined from an optimal potential vector p of D. That is, if the vectors t and f obtained from a given potential vector p by (8.12) and (8.13) satisfy conditions (8.14) and (8.15), then f and (p,t) are optimal to P and D, respectively.

To find such p, we try in the following to minimize the objective function $\xi(p)$ of (8.11). Note that the derivatives of $\xi(p)$ are given by

$$\partial\,\xi(p)/\,\partial p_i^k = \sum_{(u,v)\in A}\ \sum_{k\in K} \partial F_{uv}^{*k}(p_u^k - p_v^k - p_{uv})/\,\partial p_i^k$$

$$= \sum_{(i,m)\in A} dF_{im}^{*k}(t_{im}^k)/dt_{im}^k$$

$$- \sum_{(m,i)\in A} dF_{mi}^{*k}(t_{mi}^k)/dt_{mi}^k$$

$$= \sum_{(i,m)\in A} f_{im}^k - \sum_{(m,i)\in A} f_{mi}^k, \qquad (8.16)$$

$$\partial\,\xi(p)/\,\partial p_{ij} = \sum_{(u,v)\in A}\ \sum_{k\in K} \partial F_{uv}^{*k}(p_u^k - p_v^k - p_{uv})/\,\partial p_{ij}$$

$$+ \sum_{(u,v)\in A} \partial G_{uv}^{*}(p_{uv})/\,\partial p_{ij}$$

$$= - \sum_{k\in K} dF_{ij}^{*k}(t_{ij}^k)/dt_{ij}^k + dG_{ij}^{*}(p_{ij})/dp_{ij}$$

$$= - \sum_{k\in K} f_{ij}^k + dG_{ij}^{*}(p_{ij})/dp_{ij}, \qquad (8.17)$$

provided that we determine t_{ij}^k and f_{ij}^k by (8.12) and (8.13), respectively. We see that (8.16) and (8.17) represent the deviations from equalities (8.14) and (8.15), respectively. Denote these deviations by

$$d_i^k(p) \stackrel{\Delta}{=} \sum_{(i,m)\in A} f_{im}^k - \sum_{(m,i)\in A} f_{mi}^k \ (=\partial\,\xi(p)/\partial\,p_i^k), \qquad (8.18)$$

$$d_{ij}(p) \stackrel{\Delta}{=} - \sum_{k\in K} f_{ij}^k + dG_{ij}^{*}(p_{ij})/dp_{ij} \ (=\partial\,\xi(p)/\,\partial p_{ij}). \qquad (8.19)$$

If $d_i^k(p)$ is negative (positive) for some k and i, condition (8.16) implies that $\xi(p)$ decreases by increasing (decreasing) the corresponding potential p_i^k. Similarly if $d_{ij}(p)$ is positive (negative) for some (i,j), (8.17) implies that $\xi(p)$ decreases by decreasing (increasing) p_{ij}. In particular, if $d_i^k(p)$ and $d_{ij}(p)$ all become 0, the obtained f and (t,p) satisfy (8.12)-(8.15), and are optimal to P and D respectively. Based on this observation, we propose in the next section a framework of descent algorithms for minimizing $\xi(p)$. By specifying the details, various algorithms for solving P and D will result.

8.4 ALGORITHMS

Let e_i^k and e_{ij} be the $|K||V|+|A|$ dimensional unit vectors such that $p \cdot e_i^k = p_i^k$ and $p \cdot e_{ij} = p_{ij}$, respectively. Also let $\delta > 0$ and $\varepsilon \geq 0$ be parameters, where ε is the accuracy with which the procedure halts after finite iterations, and ε is the margin of deviations that restrict the candidates of p_i^k and p_{ij} updated in each iteration. For $\beta \geq 0$, denote

$$V_-^k(\beta) \triangleq \{i \in V \mid d_i^k(p) \leq -\beta\}, \quad V_+^k(\beta) \triangleq \{i \in V \mid d_i^k(p) \geq \beta\} \text{ for } k \in K,$$

$$A_-(\beta) \triangleq \{(i,j) \in A \mid d_{ij}(p) \leq -\beta\}, \quad A_+(\beta) \triangleq \{(i,j) \in A \mid d_{ij}(p) \geq \beta\}.$$

Procedure **MULTIFLOW**

Input: A network with upper and lower capacities $N=(G(V,A), \underline{c}, \tilde{c})$, cost functions F_{ij}^k and G_{ij} (or their conjugates F_{ij}^{*k} and G_{ij}^*), and real numbers $\delta > 0$, $\varepsilon \geq 0$.

Output: Approximate optimal solutions f of P and (t,p) of D.

Step 0(initialization): Choose an initial potential vector p and compute the corresponding t_{ij}^k and f_{ij}^k by (8.12) and (8.13), for $k \in K$, $(i,j) \in A$, as well as d_i^k and d_{ij} of (8.18) and (8.19), for $k \in K$, $i \in V$, $(i,j) \in A$. Let $k' := 0$.

Step 1(checking of the current solution): If $V_-^k(\varepsilon)=V_+^k(\varepsilon)=\emptyset$ for all $k \in K$ and $A_-(\varepsilon)=A_+(\varepsilon)=\emptyset$ (convergence check), then output the current f and (t,p) as approximate optimal solutions of P and D, respectively, and halt.

Step 2(updating δ): Choose a real number $\delta' \geq 0$ and update by $\delta := \delta'$ according to an appropriate rule. Let $k' := k'+1$ and go to Step 3.

Step 3(updating p): If $k'=K+1$ then let $k':=0$ and go to (b). Otherwise, go to (a).

(a) If $V_-^{k'}(\delta) \cup V_+^{k'}(\delta)=\emptyset$, return to Step 1. Otherwise, execute the following procedure. Choose a nonempty set $S \subseteq V_-^{k'}(\delta) \cup V_+^{k'}(\delta)$, and obtain $\Delta > 0$ and $\tilde{p}$ such that

$$\tilde{p} = p + \sum_{i \in S \cap V_-^{k'}(\delta)} \Delta\, e_i^{k'} - \sum_{i \in S \cap V_+^{k'}(\delta)} \Delta\, e_i^{k'}, \quad (8.20)$$

$$\sum_{i \in S} d_i^{k'}(\tilde{p}) = 0.$$

Update p by $p := \tilde{p}$ and return to Step 1.

(b) If $A_-(\delta) \cup A_+(\delta)=\emptyset$, return to Step 1. Otherwise, choose a nonempty set $S \subseteq A_-(\delta) \cup A_+(\delta)$, and obtain $\Delta > 0$ and $\tilde{p}$ such that

$$\tilde{p} = p + \sum_{(i,j) \in S \cap A_-(\delta)} \Delta\, e_{ij} - \sum_{(i,j) \in S \cap A_+(\delta)} \Delta\, e_{ij}. \quad (8.21)$$

$$\sum_{(i,j)\in S} d_{ij}(\tilde{p})=0.$$

Update p by $p:=\tilde{p}$ and return to Step 1. □

The conditions (8.20) and (8.21) in Step 3 intend to decrease $\xi(p)$ by modifying some components of the current potential vector p, as discussed in the last paragraph of Section 8.3. As the second condition in (8.20) or (8.21) indicates, the step size Δ is determined so that $\xi(p+\Delta\cdot e_S)$ is minimized in the direction of

$$e_S=\sum_{i\in S\cap V_-^{k'}(\delta)} e_i^{k'} - \sum_{i\in S\cap V_+^{k'}(\delta)} e_i^{k'} \text{ or}$$

$$e_S=\sum_{(i,j)\in S\cap A_-(\delta)} e_{ij} - \sum_{(i,j)\in S\cap A_+(\delta)} e_{ij},$$

respectively. The Δ may be computed by applying an appropriate one-dimensional search technique. The purpose of introducing δ and S is to faciliate the computation of each iteration by restricting the target potentials only to those that are important and easily computed.

If $S=V_-^k(0)\cup V_+^k(0)$ and $S=A_-(0)\cup A_+(0)$ are used in Steps 3(a) and (b), respectively, the potential vector p is modified in the direction of steepest descent. This method, however, requires a considerable amount of computation because all p_i^k and p_{ij} take part in the computation of Δ and $\tilde{p}$. For the

single commodity flow problem, Bertsekas [BERT 85,87] compared the following two cases: the case $|S|=1$ and the case $S=\{i\in V|\ d_i(p)<0\}$ (or $\{i\in V|d_i(p)>0\}$), where $d_i(p)$ is defined in a manner similar to (8.18). Note that the procedure using the former S corresponds to the coordinate descent method. The multicommodity versions of these cases, i.e., $|S|=1$ and $S=\{i\in V|d_i^k(p)<0\}$ (or $\{i\in V|d_i{}^k(p)>0\}$), are considered in [GALL 77] and [STER 77] respectively, though lower and upper capacities for individual commodities are not explicitly introduced therein. S in the latter case gives the maximal descent direction of $\xi(p)$ when $p_i^k, i\in V$ are modified, but much time is required for constructing the S and for updating the corresponding potentials. In fact, [BERT 85,87, STER 77] report that the former is a better selection method of S than the latter. By introducing δ, we can define a set S which lies between the above two extreme cases. To guarantee the convergence of the resulting procedures, as proved in Section 8.6, S must satisfy the condition that S always contain at least one (i,k) with $|d_i^k(p)|>\varepsilon$ ((i,j) with $|d_{ij}(p)|>\varepsilon$). The selection rule of δ must also satisfy the condition that it must eventually become $\delta\leq\varepsilon$ after finite iterations.

As described above, Procedure MULTIFLOW has much freedom as to how to determine the set S in Step 3, and as to how to update the controlling parameter δ in Step 2. We describe below some typical rules, which are used in the subsequent computational experiments.

Determination of S The following two rules are tested.

(S-1) In Step 3(a), use

$$S=V_-^k(\delta) \quad \text{and} \quad S=V_+^k(\delta)$$

alternately for the selected $k \in K$, and in Step 3(b), use

$$S=A_-(\delta) \text{ and } S=A_+(\delta)$$

alternately.

(S-2) In Step 3(a), use $S=\{i\}$ such that $i \in V_-^k(\delta) \cup V_+^k(\delta)$, and in Step 3(b), use $S=\{(i,j)\}$ such that $(i,j) \in A_-(\delta) \cup A_+(\delta)$, where the selection of i (or (i,j)) is done simply by taking the first one found during the course of checking $V_-^k(\delta) \cup V_+^k(\delta)$ (or $A_-(\delta) \cup A_+(\delta)$).

Note that (S-2) gives rise to the coordinate descent method.

Determination of δ The following three rules are tested.

(δ-1) δ = a nonnegative constant ($\leq \varepsilon$), throughout the computation.

(δ-2) $\delta = \max_{i,k,(i,j)} \{|d_i^k(p)|, |d_{ij}(p)|\}$. This rule forces to choose in Step 3 the potential p_i^k or p_{ij} with the largest deviation $|d_i^k(p)|$ or $|d_{ij}(p)|$, respectively. It is noted that if commodity kl uniquely attains the maximum, i.e., $|d_i^{kl}(p)| = \max_{i,k,(i,j)}\{|d_i^k(p)|, |d_{ij}(p)|\}$, then $V_-^{k'}(\delta)=V_+^{k'}(\delta)=\emptyset$ for $k' \neq kl$ and $A_-(\delta)=A_+(\delta)=\emptyset$ holds in Step 3(a) and (b). That is, in this case, Steps 3(a) and 3(b) are skipped until k' becomes kl. However, the computation time required for skipping Step 3 is usually negligible.

(δ-3) In Step 0, set δ initially to some $\delta_0 > \varepsilon$. We update δ only when $V_-^k(\delta)=V_+^k(\delta)=\emptyset$ for all $k \in K$ and $A_-(\delta)=A_+(\delta)=\emptyset$ hold in Step 2 (margin check). At the (i+1)-th iteration of updating in Step 2, δ' ($= \delta_{i+1}$) is determind by

$$\delta_{i+1} := \begin{cases} 0.7\,\delta_i + 0.3\tilde{d}, & \text{if } \varepsilon \leq 0.7\,\delta_i + 0.3\tilde{d} \\ \varepsilon, & \text{otherwise,} \end{cases} \qquad (8.22)$$

where

$$\tilde{d} = \{ \sum_{k \in K} \sum_{i \in V} |d_i^k(p)| + \sum_{(i,j) \in A} |d_{ij}(p)| \} / (|K||V| + |A|),$$

(i.e., the average deviation). □

The constants 0.7 and 0.3 in (δ -3) have been empirically selected on the basis of computational experiments in Section 8.5.

Combining these, we test the following six cases of MULTIFLOW in the experiments reported in the next section.

A: (S-1),(δ -1),δ =0.0.

B: (S-1),(δ -1),δ =ε .

C: (S-1),(δ -3).

D: (S-2),(δ -1),δ =ε .

E: (S-2),(δ -2).

F: (S-2),(δ -3).

Algorithm A may be regarded as a faithful approximation of the method of steepest descent, while B and C deviate from A in that those p_i^k and p_{ij} with small $|d_i^k(p)|$ and $|d_{ij}(p)|$ are neglected in choosing the potentials to be modified. Algorithms D, E and F are coordinate descent methods. Among these, E has a special property that it always chooses the potential p_i^k or p_{ij} corresponding to the largest deviation among $|d_i^k|$ and $|d_{ij}|$.

8.5 NUMERICAL RESULTS

Computation was carried out with the following test data. To obtain a feasible network with given numbers of nodes and arcs, a strongly connected network and a flow satisfying the flow conservation constraints are first generated. For this purpose, we first construct a directed Hamiltonian cycle through all given nodes, and then generate arcs between some pairs of nodes, which are selected randomly, until the network has a specified number of arcs. On the resulting graph, we construct a feasible flow for each commodity k by randomly selecting cycles in the graph, to which commodity k of the amount chosen from interval [0.0, 100.0] is assigned. The assignment to cycles is repeated until every arc has a positive flow of commodity k. After completing this procedure for all commodities k, the capacities $\tilde{c}_{ij}^k$ and $\underline{c}_{ij}^k$ are selected randomly from intervals $[f_{ij}^k, f_{ij}^k+100.0]$ and $[0.0, f_{ij}^k]$, respectively. The numbers of $\underline{c}_{ij}$ and $\tilde{c}_{ij}$ are then randomly selected from intervals $[\Sigma_k \underline{c}_{ij}^k, \Sigma_k f_{ij}^k]$ and $[\max\{\Sigma_k f_{ij}^k, \max_k \tilde{c}_{ij}^k\}, \Sigma_k \tilde{c}_{ij}^k]$, respectively. Finally all flows and capacities are scaled so that

$$\max_{k,(i,j)} \tilde{c}_{ij}^k = 100$$

holds.

All the strictly convex functions F_{ij}^k and G_{ij} in the objective function are assumed to be quadratic. Figure 1 illustrates a quadratic functions F_{ij}^k and its conjugate F_{ij}^{*k}, i.e.,

$$F_{ij}^k(x) = \begin{cases} ax^2+bx+d, & x \in [\underline{c}, \tilde{c}] \\ +\infty, & x \notin [\underline{c}, \tilde{c}], \end{cases}$$

$$F_{ij}^{*k}(y) = \begin{cases} \tilde{c}(y-b)-a\tilde{c}^2-d, & y \in [2a\tilde{c}+b, +\infty] \\ (y-b)^2/4a-d, & y \in [2a\underline{c}+b, 2a\tilde{c}+b] \\ \underline{c}(y-b)-a\underline{c}^2-d, & y \in [-\infty, 2a\underline{c}+b]. \end{cases}$$

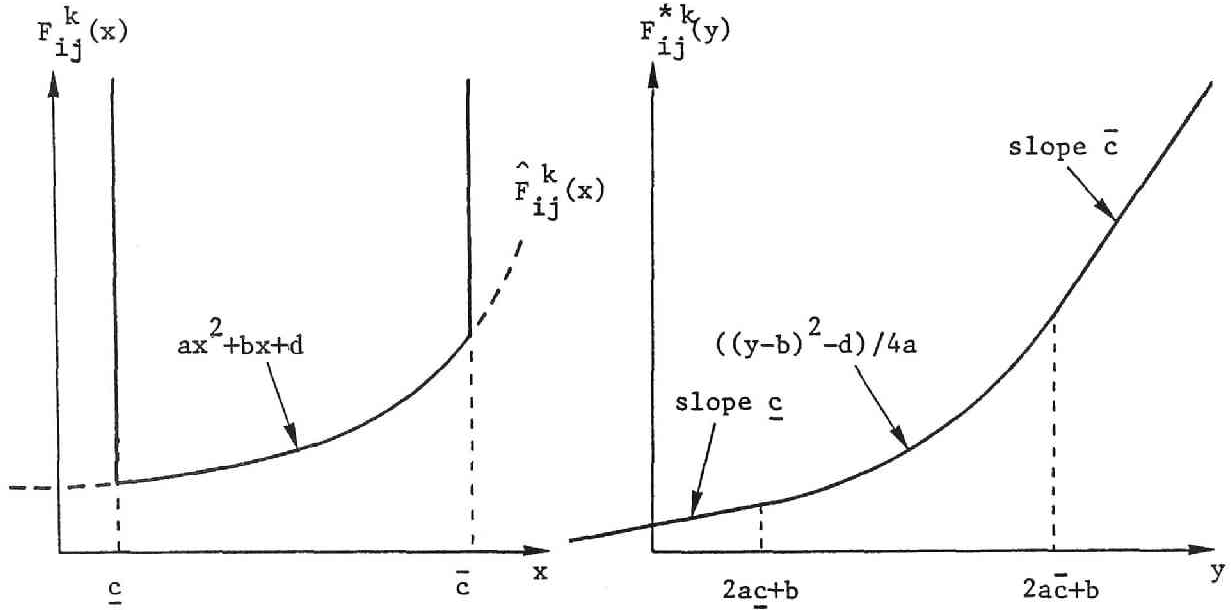
Each F_{ij}^k is defined by

$$F_{ij}^k = a(f_{ij}^k-h)^2,$$

where a is randomly chosen from interval [0.1, 10.0] and $h=((\underline{c}_{ij}^k-\tilde{c}_{ij}^k)/2)+\alpha$, where α is ramdomly chosen from interval $[-(\tilde{c}_{ij}^k-\underline{c}_{ij}^k), (\tilde{c}_{ij}^k-\underline{c}_{ij}^k)]$. Functions G_{ij} are similarly defined. In the case of quadratic functions, the step size Δ in (8.20) and (8.21) can be obtained analytically without resorting to line search techniques.

The parameter ε used in Procedure MULTIFLOW is always set to $\varepsilon=0.1$ in Figures 8.2-8.7.

The procedure MULIFLOW was coded in FORTRAN77, and run on the FACOM M-382 machine of Kyoto University Data Processing Center. Figure 8.2 shows the computation time of



(a) Primal cost function (b) Conjugate cost function

Fig. 8.1 A quadratic convex function and its conjugate.

all algorithms A-F, described in the previous section. Since it has been found that algorithm A performs much worse than others, only the results for small problems are shown. One may conclude that algorithm F is most efficient among these, and the computation time of algorithm F increases only linearly with $|A|$.

To explain the reasons for these, we illustrate in Figure 8.3 the total number of inner iterations (Step 3) for the coordinate descent methods D, E and F. (Recall that most of the computational time is spent in Step 3.) Since algorithm E always updates the potential with the largest absolute deviation, it is expected that E achieves the largest gain in one iteration, among all the coordinate descent methods. This suggests that algorithm E is most efficient if measured only by the number of iterations. This point is clearly observed in Figure 8.3. However, the computation time required for one iteration of E is much larger than others because it is quite expensive to find the potential with the largest deviation.

Figure 8.3 also indicates that the number of iterations required by E and F are almost the same. Since the computation time per iteration of F is much smaller than that of E, this explains why F is faster than E. This

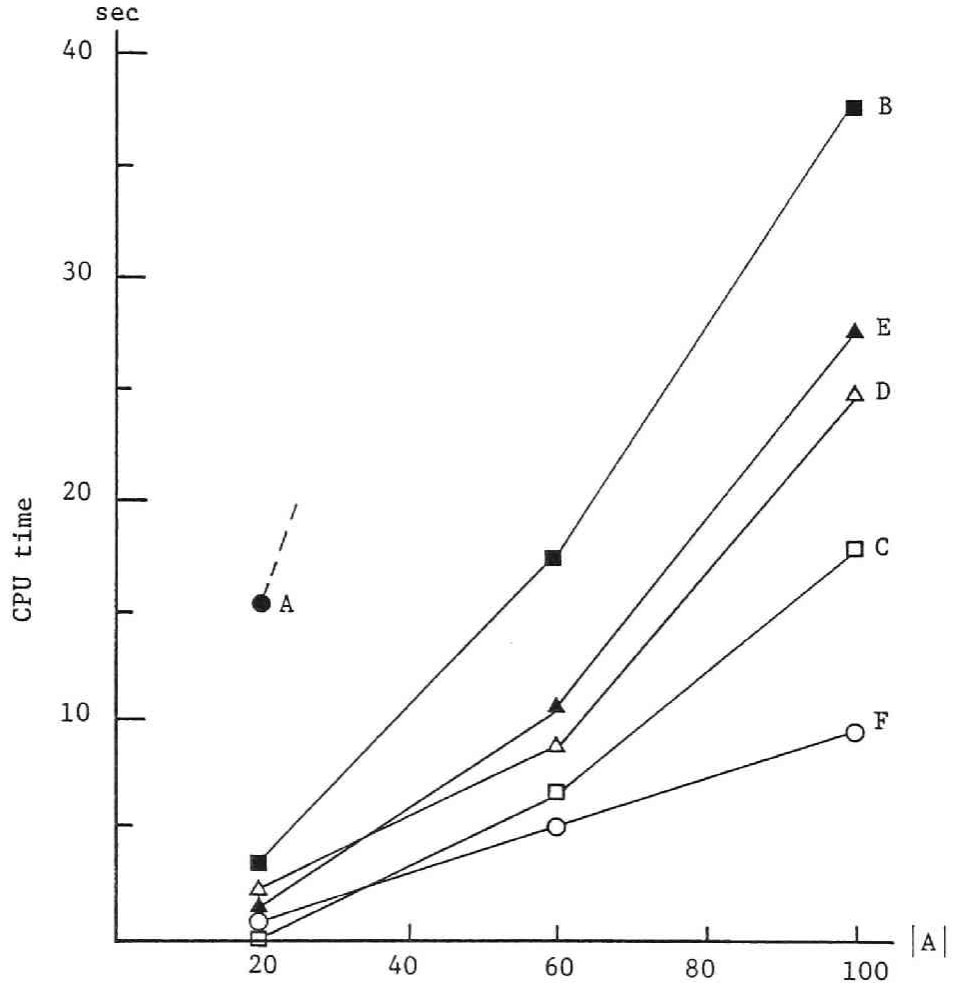


Fig. 8.2 CPU time for obtaining optimal solutions. (Each point represents the average of ten samples with $|A|=2|V|, |K|=4$.)

desirable aspect of F concerning the number of iterations may be explained as follows. When we update δ_i by rule (δ-3), absolute values of all deviations are smaller than δ_i. In most cases, it is expected that δ_i is only slightly larger than the maximum absolute deviation. As the new threshold δ_{i+1} is determined by taking into account the average $\bar{d}$ of all deviations, it is usually close to the maximum absolute deviation. For this reason, (δ-3) well simulates the behavior of (δ-2).

The constants in (8.22) are empirically selected on the basis of computational experiments as shown in Figure 8.4. We see that the optimal α in $\delta_{i+1} := \alpha\delta_i + (1-\alpha)\bar{d}$ is 0.7.

Algorithms C and F, which both adopt rule (δ-3), spend some computation time to update δ_i in Step 2. In our computational experiments, however, the total number of updating δ_i in Step 2 was always between 32 and 40, independently of the problem size. As this number is rather small compared with the number of executing Step 3, the time required for updating δ_i is usually negligible.

Figure 8.2 also shows that algorithm C gives better performance than the coordinate descent methods D and E. This means that the computational efficiency depends more on the δ-rule than on the S-rule.

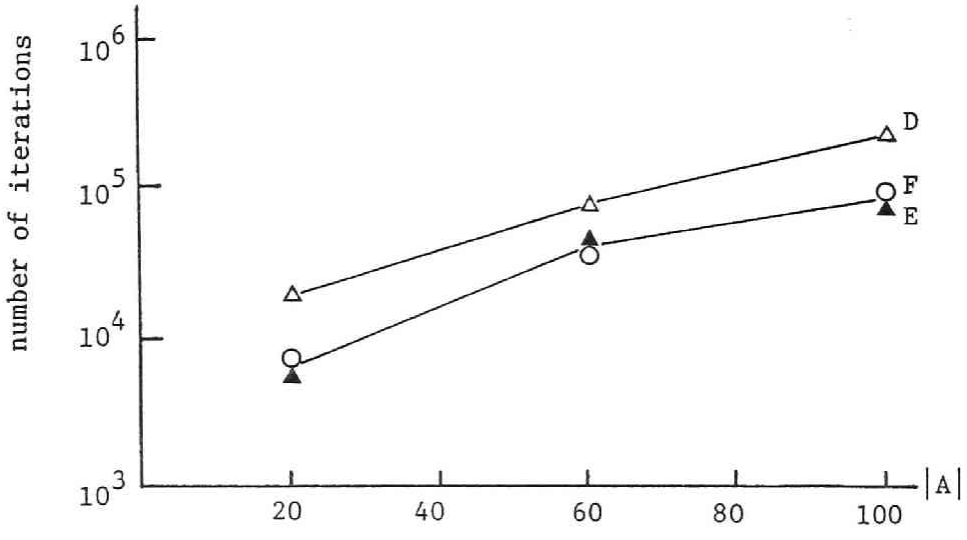


Fig. 8.3 The number of inner iterations (Step 3) (Each point represents the average of ten samples with $|A|=2|V|, |K|=4$.)

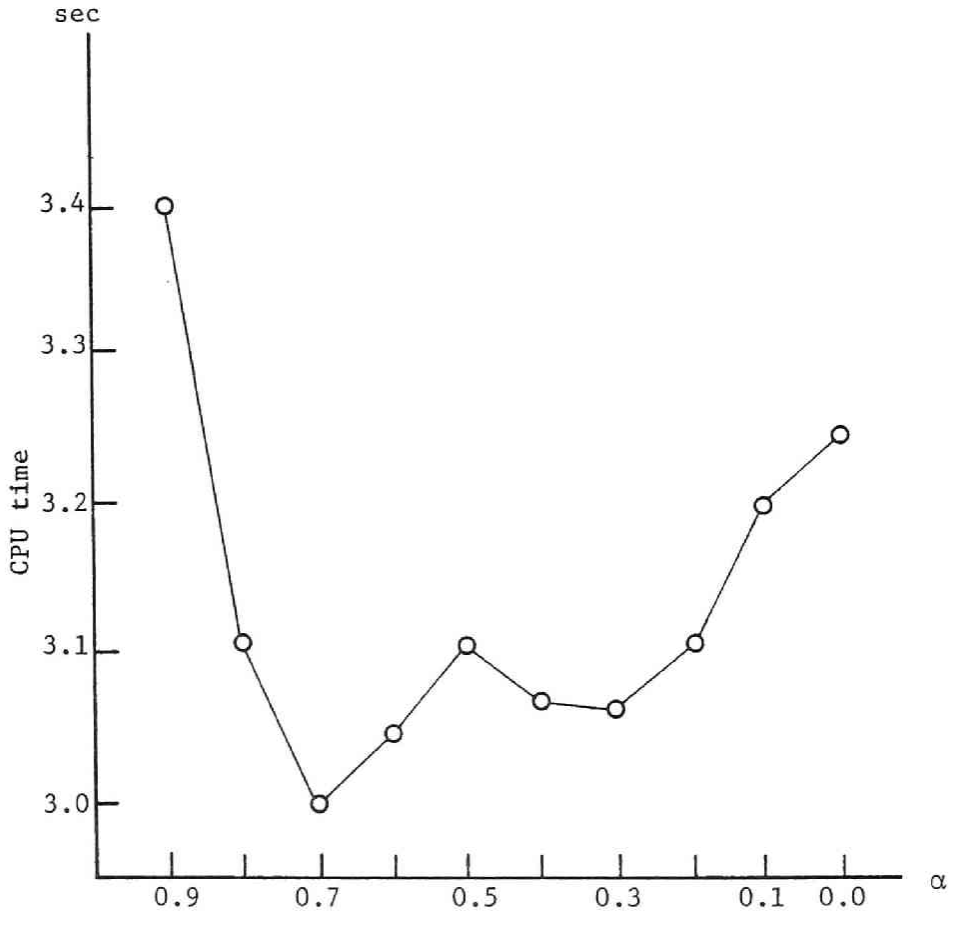


Fig. 8.4 CPU time of method F with respect to the coefficient α in(8.22). (Each point represents the average of nine samples with $|A|=150$, $|V|=50$ and $|K|=4$.)

As F appears to be most promising, we carried out further numerical tests with various network parameters $|V|$ and $|A|$. Figure 8.5 is the results for larger networks, exhibiting the behavior similar to Figure 8.2. We see that the computation time increases approximately linearly with $|A|$. Figure 8.6 shows the results of F for rather dense networks, i.e., $|A|=0.1*|V|^2$ and $|K|=7$. Even in this case, the required computation time seems to increase only linearly with $|A|$. As typical examples, F could solve problems with $|K|=7, |V|=100$ and $|A|=1000$ in about 60 seconds. Figure 8.7 illustrates the behavior of algorithm F for various values of the number of commodities $|K|$. Unfortunately, it is observed that the computation time increases rapidly with $|K|$. Finally we show in Figure 8.8 the CPU time of algorithm F versus precision ε. The convergence rate of F seems to be linear, since the computation time is approximately proportional to the inverse of $\log \varepsilon$.

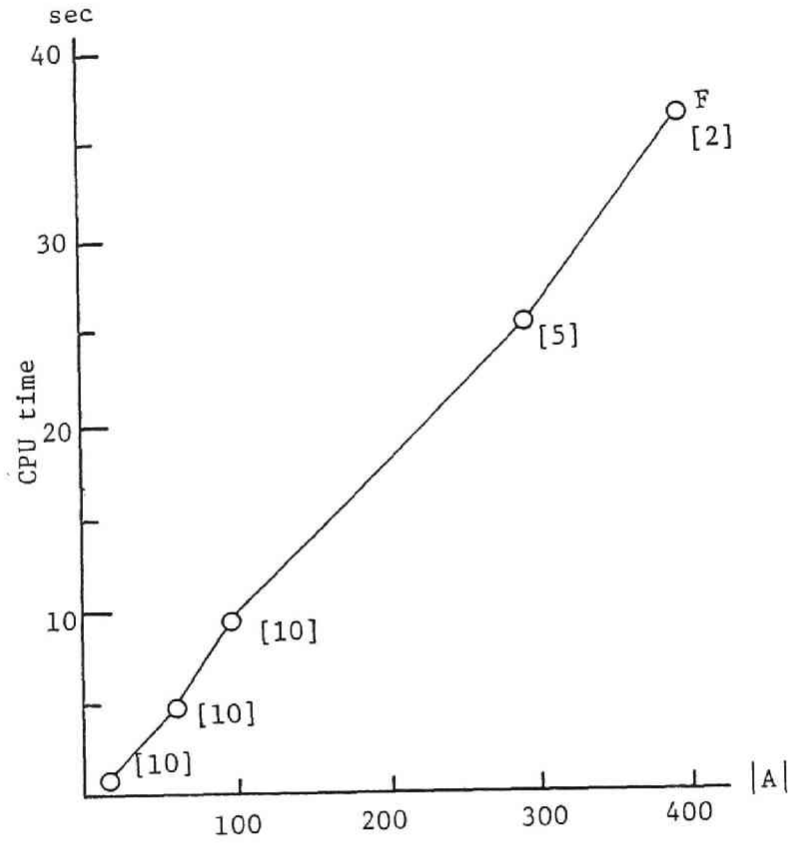


Fig. 8.5 CPU time of method F for large scale problems with $|A|=2|V|, |K|=4$ ([.]: the number of samples).

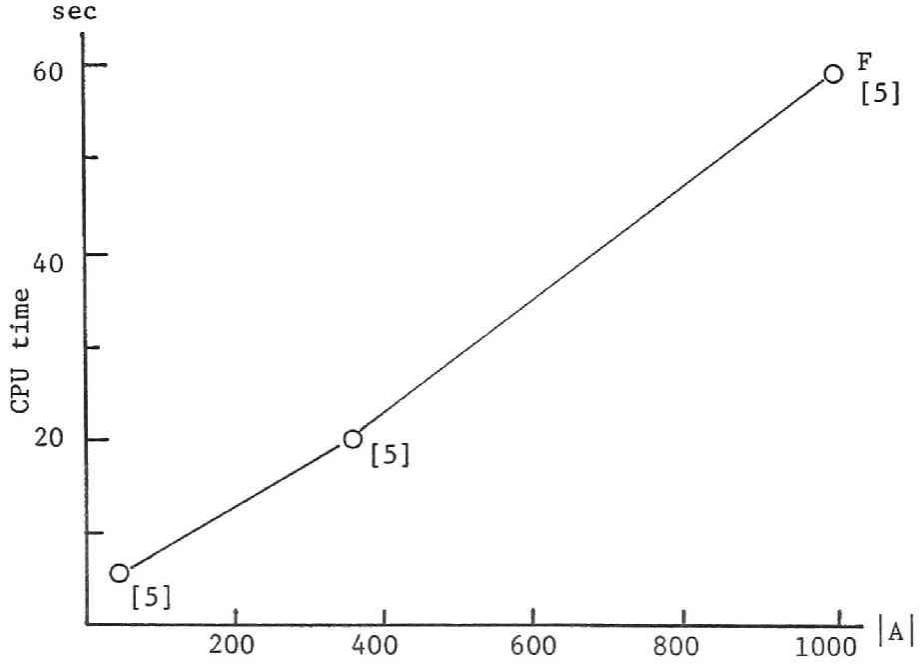


Fig. 8.6 CPU time of method F for large scale problems with $|A|=0.1|V|^2, |K|=7$ ([.]: the number of samples).

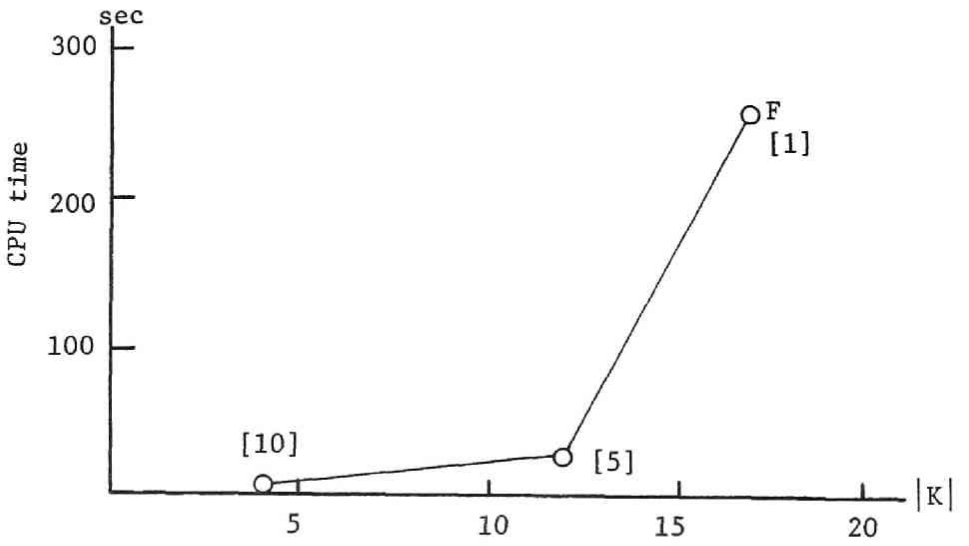


Fig. 8.7 CPU time of method F with respect to the number of commodities $|K|$ for networks with $|A|=80, |V|=20$ ([.]: the number of samples).

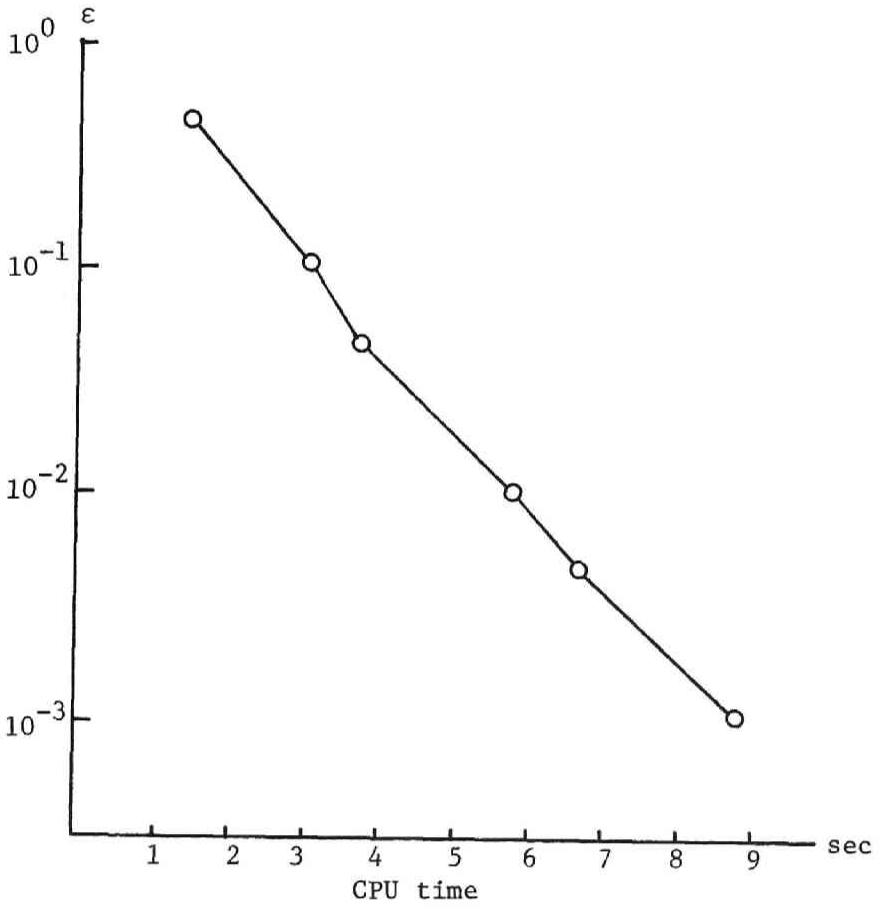


Fig. 8.8 CPU time of method F versus the precision ε.
(Each point represents the average of ten samples with $|A|=150, |V|=50, |K|=4$.)

8.6 CONVERGENCE

Before concluding this chapter, we give a proof that, for a given $\varepsilon > 0$, each of the algorithms B-F finitly terminates at an approximate optimal solution satisfying $V_-^k(\varepsilon) = V_+^k(\varepsilon) = \emptyset$ for all $k \in K$ and $A_-(\varepsilon) = A_+(\varepsilon) = \emptyset$, then output the current f, provided that the dual problem D has an optimal solution.

As noted in the last paragraph of Section 2, F_{ij}^{*k} and G_{ij}^{*} are continuously differentiable. We assume in addition that the first derivatives of $\xi(p)$ are also Lipschitz continuous, i.e., there exists a positive real number L such that

$$|| \nabla \xi(p_1) - \nabla \xi(p_2)|| \leq L \, ||p_1 - p_2||, \tag{8.23}$$

where $||x||$ is the Euclidian norm of vector x.

First let us consider the case in which a set $S \subseteq V_-^k(\delta)$ has been chosen in Step 3(a). Let vector e_S denote $\sum_{i \in S} e_i^k$. Let $\Delta(>0)$ denote the step size determined by (8.20), i.e., $d_i^k(p + \Delta e_S) = 0$. We shall estimate the amount of reduction in the dual cost $\xi(p) - \xi(p + \Delta \cdot e_S)$. For any $z \geq 0$, we have

$$\xi(p + z \cdot e_S) - \xi(p)$$

$$\leq \int_0^z \nabla\xi(p+t\cdot e_S)^T\cdot e_S \, dt \quad \text{(by the convexity of } \xi \text{)}$$

$$= \int_0^z [\, \nabla\xi(p)^T\cdot e_S + (\nabla\xi(p+t\cdot e_S) - \nabla\xi(p))^T\cdot e_S]dt$$

$$\leq z\cdot\xi(p)^T\cdot e_S + \int_0^z ||\nabla\xi(p+t\cdot e_S) - \nabla\xi(p)||\cdot||e_S||dt$$

$$\leq z\cdot\xi(p)^T\cdot e_S + L\int_0^z t||e_S||^2\, dt,$$

where the last inequality follows from (8.23). Therefore, since $||e_S||\leq|V|$ and

$$\xi(p)^T\cdot e_S = \sum_{i\in S} d_i^k(p)$$

by (8.16), we obtain

$$\xi(p) - \xi(p+z\cdot e_S) \geq -z\cdot\sum_{i\in S} d_i^k(p) - L|V|z^2/2. \qquad (8.24)$$

Moreover, since the step size Δ is determined so that the minimum of $\xi(p+z\cdot e_S)$ is attained at $z=\Delta$, it follows from (8.24) that

$$\xi(p) - \xi(p+\Delta\cdot e_S) = \max_z \{\, \xi(p) - \xi(p+z\cdot e_S)\}$$

$$\geq \max_{z} \{-z \sum_{i \in S} d_i^k(p) - L|V|z^2/2\}$$

$$= |\sum_{i \in S} d_i^k(p)|^2/(2L|V|). \qquad (8.25)$$

Similar argument applies to the other cases $S \subseteq V_+^k(\delta)$, $S \subseteq A_-(\delta)$ and $S \subseteq A_+(\delta)$ of Step 2.

Now observe that algorithms B-F are constrainted in such a way that a nonempty set S satisfying

$$|\sum_{i \in S} d_i^k(p)| \geq \varepsilon \quad \text{or} \quad |\sum_{(i,j) \in S} d_{ij}(p)| \geq \varepsilon \qquad (8.26)$$

is always selected in Step 3, by the property $\delta \geq \varepsilon$.

Under condition (8.26), the dual cost decreases at least by $\varepsilon^2/(2L|V|)$ (>0) as shown in (8.25). Therefore, if the dual problem D has an optimal solution, i.e., its optimal cost is finite, the decrease by $\varepsilon^2/(2L|V|)$ can occur only finite times. Combining this with the above argument, we conclude that algorithms B-F halt in Step 1 after a finite number of iterations, if D has an optimal solution (i.e., P has an optimal solution).

With algorithm A, however, it is not possible to guarantee (8.26), and it appears difficult to prove its finite convergence.

8.7 CONCLUSION

A framework of relaxation methods for the minimum cost multicommodity flow problem with a strictly convex objective function is presented. By specifying the rules for determining S and δ, this framework can genarate various types of procedures including the steepest descent methods and the coordinate descent methods. Limited computational experiments for the cases in which the objective function is separable and quadratic indicate that method F has the best performance among the six tested methods. The computation time of F appears to grow only linearly with the number of arcs, but grows rapidly with the number of commodities. Therefore, it is one of the future directions to develop an algorithm that is efficient for problems with many commodities.

CHAPTER 9

CONCLUSION

Throughout this thesis, we have developed efficient algorithms for the multicommodity flow problem in directed networks. The main theoretical contribution of this thesis may be classified into the following two categories.

The first category contains graph theoretic algorithms for testing feasibility of the multicommodity flow problem in certain planar directed network. Three polynomial time algorithms for classes CB, CS and CU are developed in Chapters 3, 4 and 5. Furthermore, we have shown some important properties in the network theory, e.g., the integrality of flows and the max-flow min-cut theorem: Classes CB, CS and CU have the integral flow property, and furthermore classes CB and CS have the max-flow min-cut property, while class CU does not have the latter property. We illustrate the inclusion relations among these classes in Figure 9.1. Among the known classes of the multicommodity flow problem in directed or undirected networks for which it is possible to develop efficient graph theoretic algorithms, the class of CU networks possesses a unique characteristic that it does not have the max-flow min-cut property. This

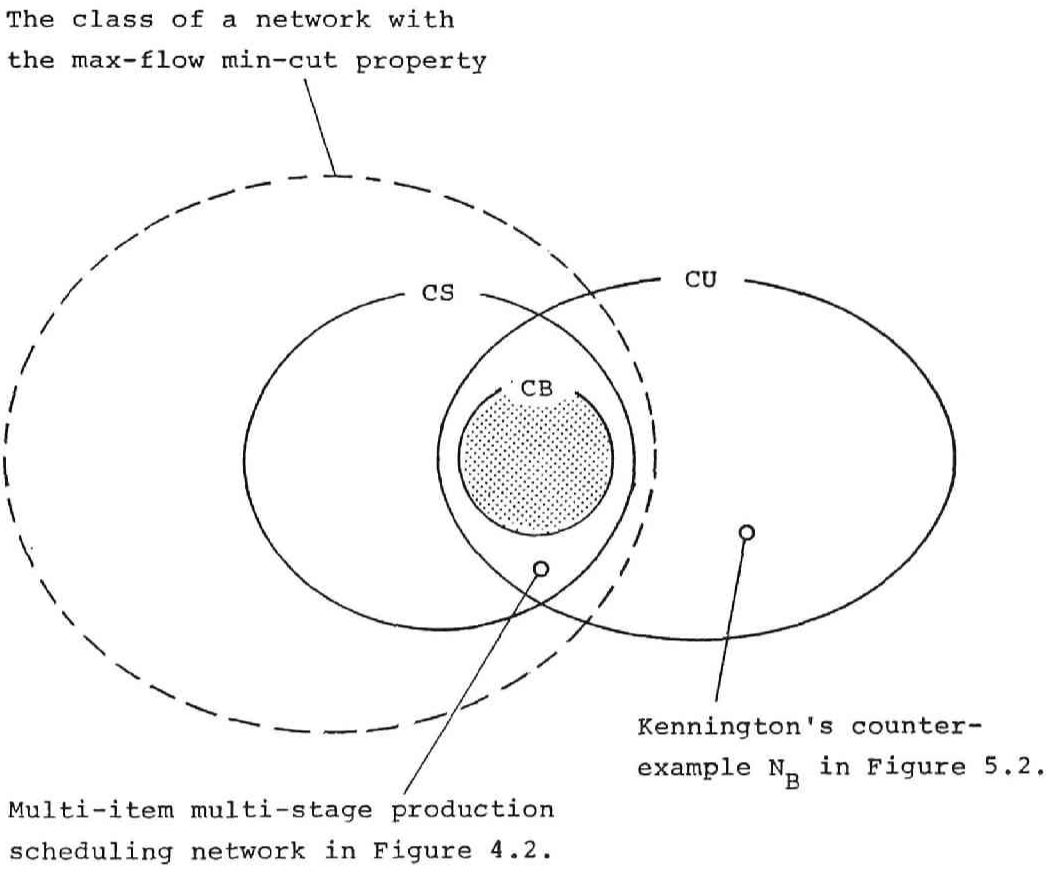


Fig. 9.1 Illustration of the inclusion relations among classes CB, CS and CU.

may suggest that the concept of capacity balance and dummy flow introduced for defining these classes will play an important role, different from the max-flow min-cut theorem, in the theory of directed networks.

In the second category, we have studied the mathematical programming approach for minimizing a nonlinear cost function of the multicommodity flow problem in a general directed network. Relaxation methods of various types for obtaining optimal solutions are proposed. Computational experiments indicate that the best one among the six tested methods can practically solve those problem which contain, for example, up to 100 nodes, 1000 arcs and 7 commodities.

The efficient algorithms proposed in the above categories would be useful and important from both theoretical and practical point of views, as many of the problems in various engineering fields can be formulated as network optimization problems. Importance of efficient algorithms in these areas will be increasing. The auther hopes that the work contained in this thesis is helpful to forward the status of network optimization techniques one step ahead.

APPENDIX

Proof of Lemma 5.4 By Lemma 5.2, N^d has a pole. Suppose that all poles of N^d, (x_i,y_i),i=1,2,...,q, are trivial. Without loss of generality, we assume that N^d has 4-alternating partitions X_i^+,Y_i^-,W_i^+, Z_i^- with $x_i \in X_i^+, y_i \in Y_i^-$ and $r(X_i^+;Y_i^-)=0$ for i=1,2,..,r, and has unilateral simple cuts $A(X_i^+)$ with $x_i \in X_i^+, y_i \in Y_i^-$ and $r(X_i^+)=0$ for i=r+1,r+2,...,q (r=q or r+1=1 are allowed).

Let f be a feasible solution of N^d. Remove all flows of the given commodities except dummy flows, i.e., for each arc $a \in A$

$$\tilde{c}(a):=c(a)-\sum_{k=1}^{K} f(a,k) \ (=f(a,d)),$$

$$\tilde{P}:=\emptyset, \ \tilde{g}^k:=0,k=1,2,...,K.$$

The resulting network $\tilde{N}^d=(G,\tilde{P},P^d,\tilde{g},g^d,\tilde{c})$ is feasible. We consider the following set of arcs in $\tilde{N}^d$, as shown in Figure A1.

$$\{b_1,b_2,...,b_m\} \triangleq \left(\bigcup_{i=1}^{r} A(X_i^+;Y_i^-)\right) \cup \left(\bigcup_{i=r+1}^{q} A(X_i^+)\right)$$

From $r(X_i^+;Y_i^-)=0$ and $r(X_i^+)=0$, we have $\tilde{c}(b_j)=0$ for each b_j. Remove $b_j=a(v_j,w_j)$,j=1,2,...,m, from $\tilde{N}^d$ and add new

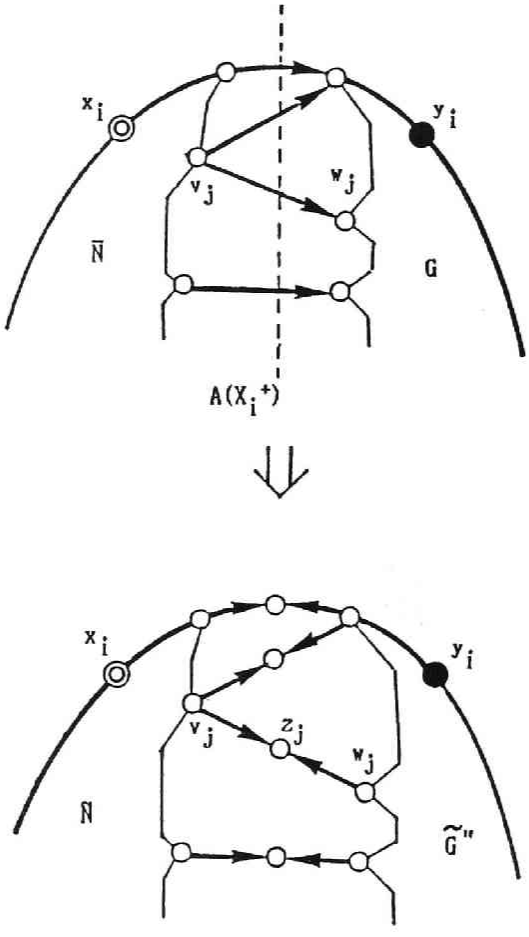


Fig. A1 Proof of Lemma 5.4.

nodes $z_j, j=1,2,\ldots,m$ and new arcs $a(v_j,z_j), a(w_j,z_j)$, $j=1,2,\ldots,m$. Define the capacities of these arcs by $\tilde{c}(a(v_j,z_j))=\tilde{c}(a(w_j,z_j))=0$, $j=1,2,\ldots,m$. Let $\tilde{G}$ be the resulting graph. Then $N^d=(\tilde{G},\tilde{P},P^d,\tilde{g},g^d,\tilde{c})$ is feasible. Each neighbouring pair (x_i,y_i) satisfies $\Pi(x_i,y_i)=\emptyset$ in G, and N^d generates no new poles. Therefore N^d has no pole, though N^d has a convergent node z_j which is not on the boundary. This contradicts Corollary 5.1. □

Lemma A1 If $\mathrm{rank}(\hat{x}_i)\leq\mathrm{rank}(\tilde{x}_i)$ holds for a pole (x_1,y_1) and an $i\in\tilde{I}$ in a feasible CU network N^d, then $\Pi(x_1,\hat{x}_i)=\Pi(x_1,\tilde{x}_i)=\emptyset$ holds. □

Proof First we assume that $\mathrm{rank}(\hat{x}_i)=\mathrm{rank}(\tilde{x}_i)$ and $\Pi(x_1,\hat{x}_i)\neq\emptyset$. See Figure A2. From $\hat{x}_i\in S^d$ and Definition 5.1 (3), $\hat{x}_i$ is unilateral. This is however impossible, because $x_1,y_i,\hat{x}_i(=\tilde{x}_i),y_{i+1}$ appear along the boundary in this order. Secondly assume $\mathrm{rank}(\hat{x}_i)<\mathrm{rank}(\tilde{x}_i)$. From the planarity of N^d, $\pi_1\in\Pi(\hat{x}_i,y_{i+1})$ and $\pi_2\in\Pi(\tilde{x}_i,y_i)$ have a common node z, as illustrated in Figure A3. Assume $\Pi(x_1,\tilde{x}_i)\neq\emptyset$, and consider (1) $z\neq\hat{x}_i,\tilde{x}_i$. Here $\pi_3\in\Pi(x_1,\tilde{x}_i)$ have a node w common to $\pi'_1\in\Pi(z,y_{i+1})$ or $\pi'_2\in\Pi(z,y_i)$. If w is in π'_1 (similarly for π'_2), then $\Pi(w,\tilde{x}_i)\neq\emptyset$ and $\Pi(\tilde{x}_i,w)\neq\emptyset$ contradicting the acyclicity of G. Therefore

$\Pi(x_1,\tilde{x}_i)=\emptyset$. We can show $\Pi(x_1,\hat{x}_i)=\emptyset$ in the same manner. The cases (2) $z=\hat{x}_i$ and (3) $z=\tilde{x}_i$ can be treated similarly. □

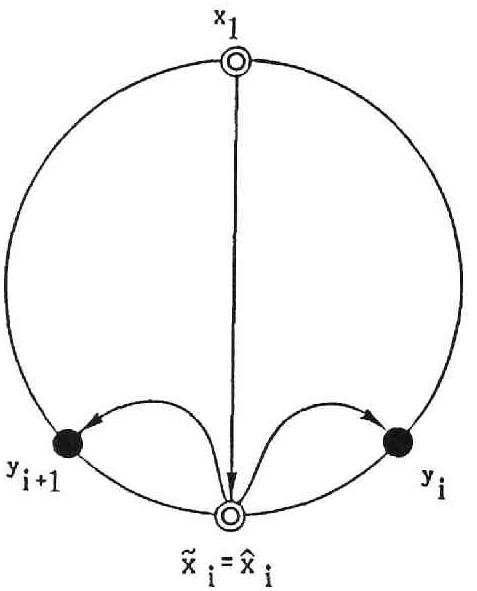


Fig. A2 Proof of Lemma A1.

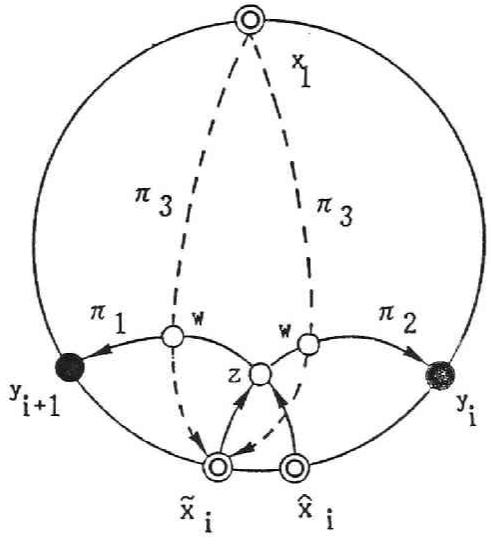


Fig. A3 Proof of Lemma A1.

Proof of Lemma 5.5 First we show that $\Pi(\tilde{x}_i, y_{i+1})=\emptyset$ holds in Case-2, i.e., there exists a convergent set $X^-(\tilde{x}_i; y_{i+1})$ by Lemma 5.1 (see Figure 5.6). Suppose $\Pi(\tilde{x}_i, y_{i+1})\neq\emptyset$, then $\mathrm{rank}(\hat{x}_i)\leq\mathrm{rank}(\tilde{x}_i)$. This means $\Pi(x_1, \tilde{x}_i)=\emptyset$ from Lemma A1 contradicting the assumption of Case-2 ($\Pi(\tilde{x}_1, x_i)\neq\emptyset$). Therefore let $X_i^+=V-X^-(\tilde{x}_i; y_{i+1})$. We obtain $x_1, y_{i+1}\in X_i^+$ by $\Pi(x_1, y_{i+1})\neq\emptyset$ and $x_i, y_1\in V-X_i^+$ by $\Pi(\tilde{x}_i, y_1)\neq\emptyset$. Therefore the boundary can be divided into the two parts contained in X_i^+ and $V-X_i^+$ respectively by Lemma 2.4.

Furthermore

$$\{z \mid \mathrm{rank}(y_1)\leq\mathrm{rank}(z)\leq\mathrm{rank}(\tilde{x}_i)\}\subseteq V-X_i^+$$

$$\{z \mid \mathrm{rank}(y_{i+1})\leq\mathrm{rank}(z)\}\cup\{x_1\}\subseteq V-X_i^+$$

are obtained. Now we assume that $A(X_i^+)$ is not an α-cut, i.e., there exist $x'\in S^d$ and $y'\in T^d$ satisfying $x'\in VB_i\cap X_i^+, y'\in V-X_i^+$ and $\Pi(x', y')\neq\emptyset$. If $\mathrm{rank}(y')<\mathrm{rank}(x_i)$ then $\pi_1\in\Pi(\tilde{x}_i, y_1)$ and $\pi_2\in\Pi(x', y')$ cross each other by the planarity, and $\Pi(x', y_1)\neq\emptyset$ holds contradicting the definition of $\tilde{x}_i$. Then we obtain $(\mathrm{rank}(y_i)<)\mathrm{rank}(\tilde{x}_i)<\mathrm{rank}(y')$ and $y'\notin T^d(x_1)$. Here if $\Pi(\tilde{x}_i, y')\neq\emptyset$ then $\Pi(x_1, y')\neq\emptyset$ by $\Pi(x_1, x_i)\neq\emptyset$ contradicting $y'\notin T^d(x_1)$. Then $\Pi(\tilde{x}_i, y')=\emptyset$ holds and there is a unilateral simple cut $A(X^+)$ with $x'\in X^+$ and $\tilde{x}_i\in V-X^+$ by Lemma 2.1. By the unilaterality of cut $A(X^+)$, we have

$x' \in X^+$. Here any path $\pi \in \Pi(x',y')$ satisfies $u \in X^+ \cap X_i^+$ and $v \in X^+ \cap X^-(\tilde{x}_i;y_{i+1})$ for some $a(u,v) \in A(\pi)$. From $v \in X^+$ $\Pi(\tilde{x}_i,v)=\emptyset$ holds. In addition, $\Pi(\tilde{x}_i,v)\neq\emptyset$ must hold for $a(u,v) \in A(V-X^-(\tilde{x}_i;y_{i+1}))$ by Lemma 2.2. This is a contradiction. Therefore the assumed x' and y' do not exist and $A(X_i^+)$ is an α-cut. □

Proof of Lemma 5.6 In Case-3, $\Pi(\tilde{x}_i,y_{i+1})=\emptyset$ holds from $\hat{S}_i^d=\emptyset$. In Case-4, if $\Pi(\tilde{x}_i,y_{i+1})\neq\emptyset$ then $\text{rank}(\hat{x}_i)\leq\text{rank}(\tilde{x}_i)$ holds and $\Pi(x_1,\hat{x}_i)=\emptyset$ by Lemma A1. This contradicts the assumption of Case-4, and hence $\Pi(\tilde{x}_i,y_{i+1})=\emptyset$. This means that a divergent set $X^+(y_{i+1};\tilde{x}_i)$ exists for Cases-3 and 4. Let $X_i^+=X^+(y_{i+1};\tilde{x}_i)$. Then similarly to Lemma 7.1, we obtain

$$\{z \mid \text{rank}(y_1)\leq\text{rank}(z)\leq\text{rank}(\tilde{x}_i)\} \subseteq V-X_i^+,$$

$$\{z \mid \text{rank}(y_{i+1})\leq\text{rank}(z)\} \cup \{x_1\} \subseteq V-X_i^+.$$

Now assume that $A(X_i^+)$ is not an α-cut, i.e., there exist $x' \in S^d$ and $y' \in T^d$ satisfying $x' \in VB_i \cap X_i^+, y' \in V-X_i^+$ and $\Pi(x',y')\neq\emptyset$. Here we show $\Pi(x',y_{i+1})=\emptyset$. In Case-3, $\Pi(x',y_{+1})=\emptyset$ from $\hat{S}_i^d=\emptyset$. In Case-4, we obtain $\text{rank}(\tilde{x}_i)<\text{rank}(y'),y' \notin T^d(x_1)$, similarly to the proof of Lemma 5.5. If $\text{rank}(\hat{x}_i)\leq\text{rank}(x')$, then $\pi_1 \in \Pi(x_1,\hat{x}_i)$ and $\pi_2 \in \Pi(x',y')$ have a common node by planarity of G. Therefore $\Pi(x_1,y')\neq\emptyset$ holds. This contradicts $y' \notin T^d(x_1)$,

and we have $\mathrm{rank}(x')<\mathrm{rank}(\hat{x}_i)$. Therefore $\Pi(x',y_{i+1})=\emptyset$ in Case-4, since $\Pi(x',y_{i+1})\neq\emptyset$ contradicts the definition of $\hat{x}_i$.

By $\Pi(x',y_{i+1})=\emptyset$ and Lemma 2.1, there exists an unilateral simple cut $A(V-X^-)$ with $x'\in X^-$ and $y_{i+1}\in V-X^-$. From the unilaterality of cut $A(V-X^-)$, $V(\pi)\subseteq X^-$, $A(\pi)\cap A(X_i^+)\neq\emptyset$ holds for any path $\pi\in\Pi(x',y')$. Here $u\in X^-\cap X_i^+$ and $v\in V-X^+$ for some $a(u,v)\in A(X_i^+)$. From the convergency of X^-, it holds $\Pi(u,y_{i+1})=\emptyset$. This, however contradicts the result $\Pi(u,y_{i+1})\neq\emptyset$ obtained by applying Lemma 5.2 to $X^+(y_{i+1};\bar{x}_i)$. Therefore the assumption is false, and $A(X_i^+)$ is an α-cut. □

Proof of Lemma 5.7 Based on $V_i^+=\emptyset$, we first show $\mathrm{rank}(\bar{x}_i)<\mathrm{rank}(\hat{x}_i)$. As $\mathrm{rank}(\hat{x}_i)\neq\mathrm{rank}(\bar{x}_i)$ is clear, assume that $\mathrm{rank}(\hat{x}_i)<\mathrm{rank}(\bar{x}_i)$. By Lemma 2.4, there is an arc $a(u,v)$ such that $a(u,v)\in A(V-X^-(x_1;\hat{x}_i))\cap A_B$ and $\mathrm{rank}(\hat{x}_i)\leq\mathrm{rank}(u)<\mathrm{rank}(v)<\mathrm{rank}(\bar{x}_i)$. Applying Lemma 5.1 to $X^-(x_1;\hat{x}_i)$, we obtain $\Pi(x_1,z)\neq\emptyset$ (see Figure A4). Here by $\Pi(x_1,y_i)\neq\emptyset$ and the planarity, $\Pi(\bar{x}_i,y_i)\neq\emptyset$ holds, and $\pi_1\in\Pi(\bar{x}_i,y_i)$ and $\pi_2\in\Pi(\hat{x}_i,y_{i+1})$ have a common node w. By the unilaterality of the cut, w is not contained in $(V-X^-(x_1;\bar{x}_i))\cap(V-X^-(x_1;\hat{x}_i))$. Now assume that $\Pi(x_1,w)\neq\emptyset$

and $\Pi(w,z)\neq\emptyset$ hold or w=z holds. (Otherwise, based on a path $\pi_3 \in \Pi(x_1,z)$ that does not pass through node w, we can find a directed cycle in a manner similar to the proof of Lemma A1.) Here if $w \notin V_B$ then w is unilateral from the

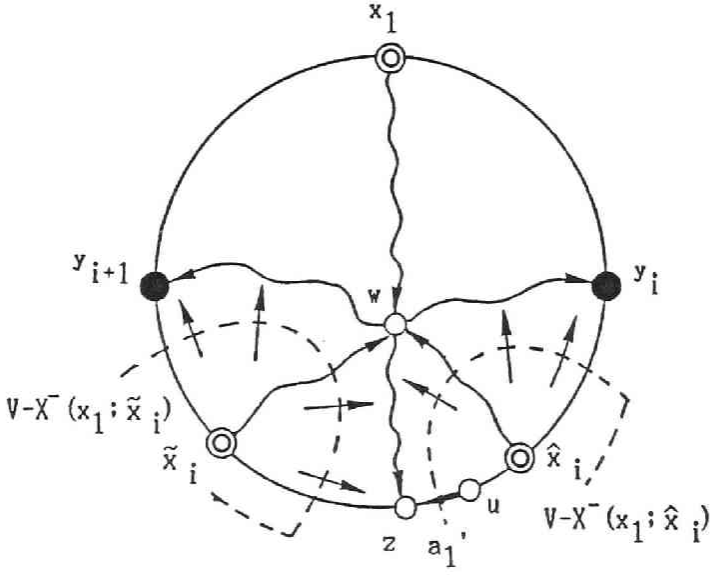


Fig. A4 Proof of Lemma 5.7.

definition of a CU network. This, however is impossible as obvious from Figure A4. In case $w \in V_B$, we can derive a contradiction by the same argument, and then obtain $\text{rank}(\tilde{x}_i)<\text{rank}(\hat{x}_i)$.

If $\Pi(\tilde{x}_i, y_{i+1})\neq\emptyset$, then $\text{rank}(\tilde{x}_i)<\text{rank}(\hat{x}_i)$ contradicts the definition of $\hat{x}_i$. Therefore $\Pi(\tilde{x}_i, y_{i+1})=\emptyset$, and there exists $X^+(y_{i+1};\tilde{x}_i)$. Let $X_i^+=X^+(y_{i+1};\tilde{x}_i) \cup (V-X^-(x_1;\hat{x}_i))$. By Lemma 2.4, there exist four arcs a_1, a_2, b_1, b_2 such that $A(X^+(y_{i+1};\tilde{x}_i)) \cap A_B=\{a_1, a_2\}$ and $A(V-X^-(x_1;\hat{x}_i)) \cap A_B=\{b_1, b_2\}$, where the ranks of the end points of a_1 (b_1) are larger than those of the end points of a_2 (b_2). Here $\hat{x}_i \in X^+(y_{i+1};\tilde{x}_i)$ from $\Pi(\hat{x}_i, y_{i+1})\neq\emptyset$. Both $X^+(y_{i+1};\tilde{x}_i)$ and $V-X^-(x_1;\hat{x}_i)$ contain $\hat{x}_i$ but not $\tilde{x}_i$. Let $a_2=a(z,z')$, $b_2=a(w,w')$. Then

$\text{rank}(\tilde{x}_i)<\text{rank}(z')<\text{rank}(z)\leq\text{rank}(\hat{x}_i)$, and

$\text{rank}(\tilde{x}_i)<\text{rank}(w')<\text{rank}(w)\leq\text{rank}(\hat{x}_i)$

hold.

Now we show $A(X_i^+)$ is a unilateral simple cut and furthermore is an α-cut by considering two cases (1) $\text{rank}(w)\leq\text{rank}(z)$ (see Figure 5.8), and (2) $\text{rank}(z)<\text{rank}(w)$ (see Figure A5). First consider case (1). As two connected divergent sets $X^+(y_{i+1};\tilde{x}_i)$ and $V-X^-(x_1;\hat{x}_i)$ both contain $\hat{x}_i$, their union X_i^+ is a connected divergent set. Since all nodes whose rank is larger than $\text{rank}(\hat{x}_i)$ are

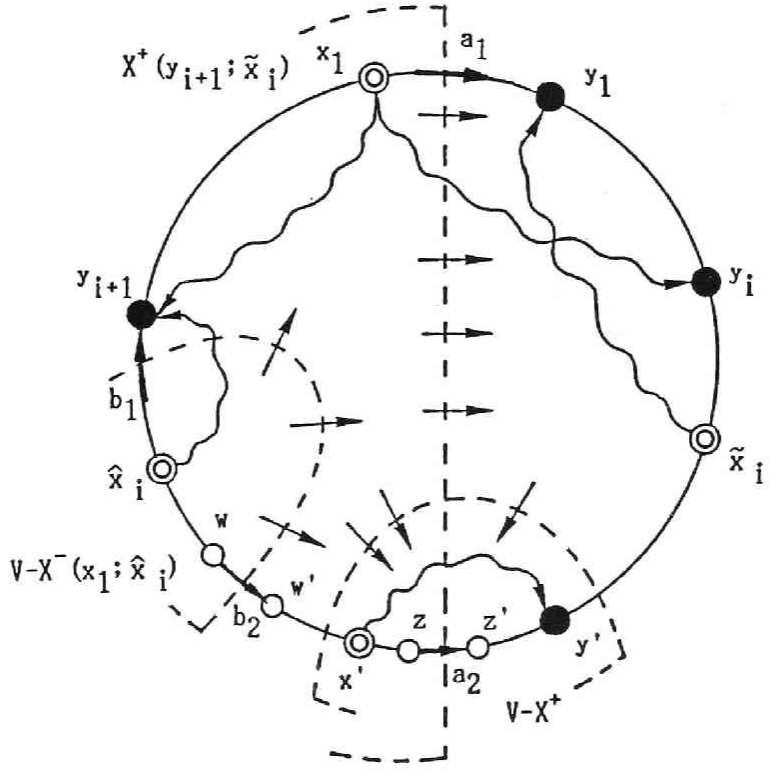


Fig. A5 Proof of Lemma 5.7.

contained in $X^+(y_{i+1};\tilde{x}_i)$ and the ranks of the end points of b_1 are larger than that of $\hat{x}_i$, $X^+(y_{i+1};\tilde{x}_i)$ contains the end ponits of b_1. Similarly $V-X^-(x_1;\hat{x}_i)$ contains the end points of a_2, or $a_2=b_2$ holds. Therefore we have $A(X_i^+)\cap A_B=\{a_1,b_2\}$. By Lemma 2.4, this means that convergent set $V-X^-(x_1;\hat{x}_i)$ is connected. Therefore, unilateral cut $A(X_i^+)$ is simple. In case (2), a similar argument can be used to show that $A(X_i^+)$ is simple.

Secondly, We show that $A(X_i^+)$ is an α-cut. For this, assume $A(X_i^+)$ is not an α-cut. Here, from the unilaterality of $A(X_i^+)$, we have easily

$$\{z\,|\,\mathrm{rank}(y_1)\leq\mathrm{rank}(z)\leq\mathrm{rank}(\tilde{x}_i)\}\in V-X_i^+,$$

$$\{z\,|\,\mathrm{rank}(y_{i+1})\leq\mathrm{rank}(z)\}\cup\{x_1\}\in X_i^+.$$

Therefore, this assumption implies that there exist $x'\in S^d$ and $y'\in T^d$ satisfying $x'\in VB_i\cap X_i^+, y'\in V-X_i^+$ and $\Pi(x',y')\neq\emptyset$. First we consider case (1). From $y'\in V-X_i^+$, we obtain $\mathrm{rank}(y')<\mathrm{rank}(y_{i+1})$. If $\mathrm{rank}(y')<\mathrm{rank}(\tilde{x}_i)$, then $\Pi(\tilde{x}_i,y_i)\neq\emptyset$ and $\Pi(x',y')\neq\emptyset$ mean $\Pi(x',y_1)\neq\emptyset$ by the palanarity. This, however, contradicts the definition of $\tilde{x}_i$, and we have $\mathrm{rank}(\tilde{x}_i)<\mathrm{rank}(y')<\mathrm{rank}(\hat{x}_i)$, that is, $y'\notin T^d(x_1)$ and $\Pi(x_1,y')=\emptyset$. Here, by Lemma 2.1, there is a unilateral simple cut $A(X_i^+)$ such that $y'\in X^+$ and $x_1\in V-X^+$. From the divergency of X^+ and $\Pi(x',y')\neq\emptyset$, we have $x'\in X^+$. Since

each of X^+ and $V-X^+$ contains one of the two parts of the boundary, nodes w,w' (the end points of b_2) are contained in X^+. Then $\Pi(x_1,w')=\emptyset$ from the divergency of X^+. This, however contradicts $\Pi(x_1,w')=\emptyset$ that is obtained by applying Lemma 5.1 to $X^-(x_1;\hat{x}_i)$. $A(X_i^+)$ is an α-cut.

Now we consider case (2). Similarly for case (1), we have $\mathrm{rank}(\tilde{x}_i)<\mathrm{rank}(y')<\mathrm{rank}(\hat{x}_i)$. If $\mathrm{rank}(w')\leq\mathrm{rank}(x')$, then we have a unilateral simple cut $A(X_i^+)$ with $y'\in X^+$ and $x_1\in V-X^+$ by $\Pi(x_1,y')=\emptyset$ and Lemma 4.1, as shown in a manner similar to case (1). Hence w' (the terminal node of b_2) satisfies $w'\in X^+$, that is, $\Pi(x_1,w')=\emptyset$. This, however, contradicts $\Pi(x_1,w')=\emptyset$ that is obtained by applying Lemma 4.2 to $X^-(x_1;\hat{x}_i)$. This implies $\mathrm{rank}(x')<\mathrm{rank}(w')$. Also $x'\neq\hat{x}_i$ and $\Pi(x',y_{i+1})=\emptyset$ hold by $x'\in V-X^-(x_1;\hat{x}_i)$. Then there is a unilateral simple cut $A(X^+)$ with $y_{i+1}\in X^+$ and $x'\in V-X^+$. Furthermore $y'\in V-X^+$ holds by its unilaterality. As cut $A(X^+)$ is unilateral, $V(\pi)\subseteq V-X^+$ and $A(\pi)\cap A(X^+(y_{i+1};\tilde{x}_i))\neq\emptyset$ hold for any path $\pi\in\Pi(x',y')$. Then $\Pi(u,y_{i+1})=\emptyset$ for some $a(u,v)\in A(X_i^+)$. This, however contradicts the result $\Pi(u,y_{i+1})\neq\emptyset$ that is obtained by applying Lemma 4.2 to $X^+(y_{i+1};\tilde{x}_i)$. Therefore $A(X_i^+)$ is also an α-cut in case (2). □

Proof of Lemma 5.8 We first show $X^-(x_1;\tilde{x}_i)=X^-(x_1;\hat{x}_i)$ from the assumption that $V_i^+\neq\emptyset$ in Case-6. If we assume $w \in (V-X^-(x_1;\hat{x}))-(V-X^-(x_1;\tilde{x}_i))$, there exists an arc $a(w',w)$ with $w' \in V_i^+$ by the connectivity of $V-X^-(x_1;\hat{x}_i)$. Then

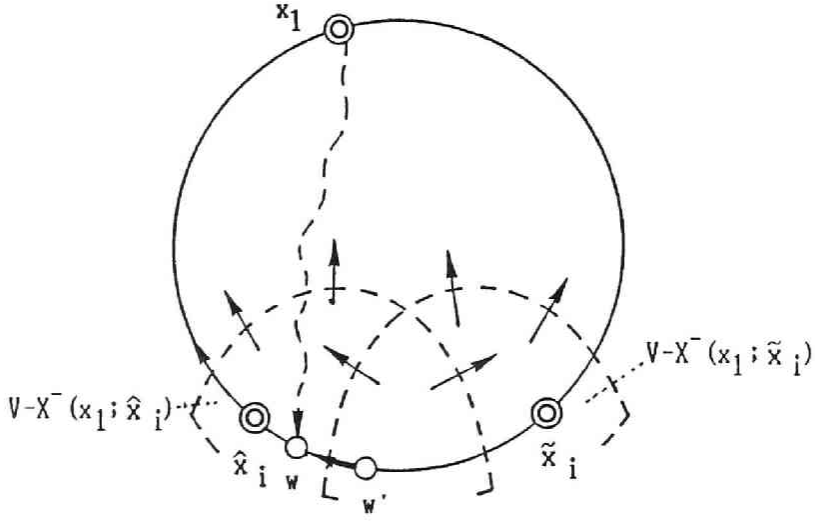


Fig. A6 Proof of Lemma 5.8.

$\Pi(x_1,w)\neq\emptyset$ by Lemma 5.1 contradicting the convergency of $X^-(x_1;\hat{x}_i)$. Therefore $V-X^-(x_1;\hat{x}_i)\subseteq V-X^-(x_1;\tilde{x}_i)$. Similar argument can show the reverse inclusion-relation. Now shrink the connected divergent set V_i^+ into a node w_0. By Lemma 2.6, the resulting graph G" is also CU. Here the four nodes x_1,y_1,w_0,y_{i+1} are 4-reachable, and G" has a 4-alternating partition such that $x_1\in X_i^+$, $y_1\in Y_i^-$, $w_0\in\hat{W}_i^+$, $y_{i+1}\in Z_i^-$ by Lemma 5.3. From $\Pi(x_1,y_i)\neq\emptyset$ and $\Pi(w_0,y_i)\neq\emptyset$, we obtain $y_i\in Y_i^-$ and $y_1,y_2,\ldots,y_i\in Y_i^-$ by Lemma 2.4. $A(\hat{W}_i^+)$ is a simple cut even in G by Lemma 2.6. Let $W_i^+=(\hat{W}_i^+-\{w_0\})\cup V_i^+$ then $\tilde{x}_i,\hat{x}_i\in W_i^+$. Consequently, the existence of a β-partition X_i^+,Y_i^-,W_i^+,Z_i^- in G is shown. □

Proof of Lemma 5.9 Compute $A(X_i^+)$ in the decreasing order of $i\in\tilde{I}$, and shrink X_i^+ before computing α or β-cut for the next $i\in\tilde{I}$. Since the required time to compute the maximal directed tree for an i is $O(|V|)$, the entire time is $O(|V|^2)$. □

Proof of Lemma 5.10 For simplicity, we consider the case in which both i and j have α-cuts A_i,A_j (see Figure A7). The other cases can be shown similarly. If $r_i=0$, the lemma is obvious. Therefore, assume $r_i>0$. From the definition of α-cut $A(X_i^+)$, the source x" and the sink y" for any dummy

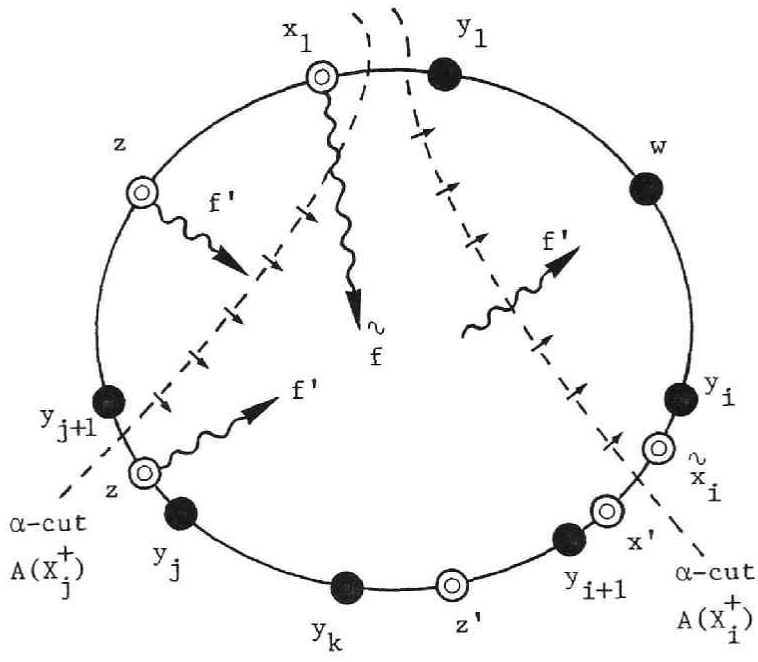


Fig. A7 Proof of Lemma 5.10.

flow passing through $A(X_i^+)$ satisfy

$\mathrm{rank}(y_{i+1}) \leq \mathrm{rank}(x'')$, and $\mathrm{rank}(y_1) \leq \mathrm{rank}(y'') \leq \mathrm{rank}(y_i)$, (otherwise $x'' \in \tilde{S}_i^d$ or $y'' \in T^d(x_1)$ are easily shown contradicting the definition of $\tilde{x}_i$ or $T^d(x_1)$). If $e_i < \min\{e_j, r_i\}$, then some part of dummy flow f from source x_1 passing through A_j enters dummy sinks $y_{i+1}, y_{i+2}, \ldots, y_j$ contained in $X_i^+ - X_j^+$. By $e_i < r_i$, there exists another dummy flow f' passing through A_j. We consider the source of this f'. Since $\Pi(x', y') = \emptyset$ for any $x' \in VB_i \cap X_i^+ \cap S^d$ and $y' \in (V - X_i^+) \cap T^d$ as a property of an α-cut, the source of f' is not in $VB_i \cap X_i^+$. Now we show that the source of f' can not be equal to any node $z' \in S^d$ with $\mathrm{rank}(y_{i+1}) < \mathrm{rank}(z') < \mathrm{rank}(y_j)$ (see Figure A7). Suppose some dummy flow from such z' enters $w \in (V - X_i^+) \cap T^d$. Since any directed path from z' to w and any path in $\Pi(x_1, y_{i+1})$ cross each other by the planarity, it holds $\Pi(x_1, w) \neq \emptyset$. Therefore $w \in T^d(x_1)$. Thus w is equal to one of $y_1, y_2, \ldots, y_i$, and $\Pi(z', y_1) \neq \emptyset$ holds because any path from z' to w and any path in $\Pi(x_1, y_1)$ cross each other. This contradicts the assumption that $S_k^d = \emptyset$ holds for any k with $i < k < j$. Therefore the source z of f' satisfies $\mathrm{rank}(y_j) < \mathrm{rank}(z)$. Thus any path π_1 of dummy flow f from x_1 to y_k $(i+1 \leq k < j)$ and any path π_2 of dummy flow f' from $z \in S^d$ to $w \in T^d \cap (V - X_i^+)$ cross each other again by the

planarity. Then we exchange the flows f and f' in paths π_1 and π_2 in the same manner as discussed in the proof to Lemma 3.1. Repeating this modification, the amount e_i of f through A_i will eventually become equal to $\min\{e_j, r_i\}$. □

Proof of Lemma 5.11 We first show that there exists a feasible flow in which the amount e_i of a dummy flow from source x_1 passes through A_i for each $i \in \tilde{I}$, where $e_i=\min\{g^d_{x1}, r_{q-1}, r_{q-2}, \ldots, r_i\}$. Let i" ($i^*$) be the maximun (minimum) i in $\tilde{I}$.

For $i'' \in \tilde{I}$, we consider a dummy flow whose amount is $e_{i''}=\min\{g^d_{x1}, r_{i''}\}$. Assume $r_{i''}>0$, since otherwise the network is infeasible. Now we assume that the amount of dummy flow passing from source x_1 through $A_{i''}$ is less that $e_{i''}$. Then there is a dummy flow f' from source $x'(\neq x_1)$ passing through $A_{i''}$ and dummy flow $\tilde{f}$ from source x_1 not passing through A_i. These f' and $\tilde{f}$ cross each other by the planarity as shown next. By definition of an α-cut (or β-semi-cut) $A_{i''}$, source $x'(\neq x_1)$ of f' satsifies $\mathrm{rank}(y_{i''+1})<\mathrm{rank}(x')$, as shown in Figure A8. Let y' be a sink of f'. If $\mathrm{rank}(x')<\mathrm{rank}(y_q)$, then a path in $\Pi(x', y')$ and a path in $\Pi(x_1, y_{i''+1})$ have a common node and therefore $\Pi(x_1, y')\neq\emptyset$. That is, $y' \in T^d(x_1)$ and y' is one of $y_1, y_2, \ldots, y_{i''}$. This

means rank(y')$\leq$rank($y_{i"}$). Here $\Pi(x',y_1)\neq\emptyset$ holds, however, because there exists $\tilde{x}_{i"}$ with $\Pi(\tilde{x}_{i"},y_1)\neq\emptyset$ from $\tilde{S}^d_{i"}\neq\emptyset$, and a path in $\Pi(\tilde{x}_{i"},y_1)$ and a path in $\Pi(x',y')$ have a common node. This contradicts the definition of i". Consequently

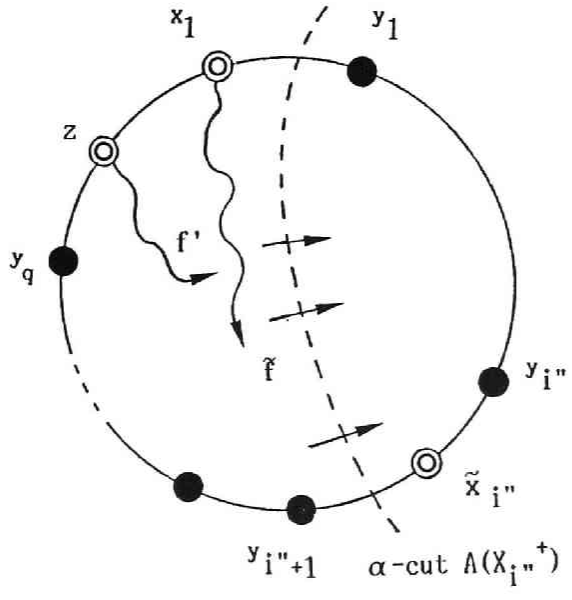


Fig. A8 Proof of Lemma 5.11.

$\mathrm{rank}(y_q)<\mathrm{rank}(x')$ holds, and f' and $\tilde{f}$ cross each other by planarity. We now exchange flows f' and $\tilde{f}$ in the same manner as in the proof for Lemma 5.10. As a result of this exchange, we obtain the amount e_i'' of dummy flow from source x_1 that passes through A_i''.

For each $j' \in \tilde{I}$, we show that any $i \in \tilde{I}$ with $j' \leq i$ has a feasible flow in which the amount e_i of dummy flow from source x_1 passes through A_i, where $e_i=\min\{g^d_{x1}, r_{q-1}, r_{q-2}, \ldots, r_i\}$. For $j'=i''$, this is obviously true. Now consider i' ($<j'$) such that $k \notin \tilde{I}$ for all k with $i'<k<j'$. By Lemma 5.10, there exists a feasible flow in which the amount $e_{i'}$ of a dummy flow from source x_1 passes through $A_{i'}$, where $e_{i'}=\min\{e_{j'}, r_{i'}\}(=\min\{g^d_{x1}, r_{q-1}, r_{q-2}, \ldots, r_i\})$. By using induction, we can show that, from $j'=i''$ to $j'=i^*$, there is a feasible flow in which the amount $e_{j'}$ of dummy flow from source x_1 passes through $A_{j'}$, where $e_{j'}=\min\{g^d_{x1}, r_{q-1}, r_{q-2}, \ldots, r_j\}$.

Finally we consider the amount of dummy flow from source x_1 to sink y_1 that passes through A_i^*. Let f^* be the dummy flow from source x_1 that passes through A_i^*. From the previous argument, we have a feasible flow f^* whose amount is e_i^*. Assume that A_i^* is an α-cut (as the case of a β-semi-cut can be similarly treated). Assume that the

amount of dummy flow in f^* to sink y_1 is less than $\min\{e_i^*, g^d_{y_1}\}$. In this case, we have a dummy flow $\tilde{f}$ to a sink ($\neq y_1$) and a dummy flow f' from some source z to sink y_1. Clearly $\mathrm{rank}(y_i^*) < \mathrm{rank}(z)$ holds by definition of i^*. Then we may have the $\min\{e_i^*, g^d_{y_1}\}$ amount of dummy flow from source x_1 to sink y_1 by modifying flows $\tilde{f}$ and f' in the same manner as in the proof for Lemma 5.10. □

REFERENCES

[AHO 74] Aho,A.V., Hopcroft,J.E. and Ullman,J.D.: The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass. (1974).

[AHO 83] Aho,A.V., Hopcroft,J.E. and Ullman,J.D.: Data Structure and Algorithms, Addison-Wesley, Reading, Mass. (1983).

[BELL 57] Bellman,R.E.: Dynamic Programming, Princeton University Press, Princeton N.J. (1957).

[BERG 73] Berge,C.: Graphs and Hypergraphs, North-Holland Publishing (1973).

[BERT 85] Bertsekas,D.P.: "A unified framework for primal-dual methods in minimum cost network flow problems", Math. Prog., Vol.32, pp.125-145 (1985).

[BERT 87] Bertsekas,D.P., Hossein,P.A. and Tseng,P.: "Relaxation methods for network flow problems with convex arc costs", SIAM J. Control and Optimization, Vol.25, pp.1219-1243 (1987).

[DIAZ 72] Diaz,H. and de Ghellink,G.: "Multicommodity maximam flow in plannar networks (The D-algorithm approach)", CORE discussion paper

No.7212, Center for Operations Research and Econometrics, Louvain-la-Neuve, Belgium (1972).

[DINI 70] Dinic,E.A.: "Algorithm for solution of a problem of maximal flow in a network with power estimation", Soviet Math. Dokl., Vol.11, pp.1277-1280 (1970).

[DREY 77] Dreyfus,S.E. and Law,A.M.: The Art and Theory of Dynamic Programming, Academic Press, New York (1977).

[EDMO 72] Edomonds,J. and Karp,R.M.: "Theoretical improvements in the algorithmic efficiency for network flow problems", J. ACM, Vol.19, pp.248-264 (1972).

[FORD 62] Ford,L.R. and Fulkerson,D.R.: Flows in Networks, Princeton University Press, Princeton, N,J. (1962).

[FRED 83] Frederickson,G.N.: "Shortest path problems in planar graphs", Proc. 24-th Symposium on Foundation of Computer Science, Tucson, Nov. pp.242-247 (1983).

[FUJI 86] Fujishige,S.: "A capacity-rounding algorithm for the minimum-cost circulation problem: a dual framework of the Tardos algorithm", Math.

Prog., pp.298-308 (1986).

[FUKU 84] Fukushima,M.: "A nonsmooth optimization approach to nonlinear multicommodity network flow problems", Journal of the Operation Research Society of Japan, Vol.27, No.2, pp.151-177 (1984).

[GALI 86] Galil,Z. and Tardos,E.: "An $O(n^2(m+n\log n)\log n)$ min-cost flow algorithm", Proc. 27-th Annual IEEE Symp. on Foundations of Computer Science, pp.1-9 (1986).

[GALL 77] Gallager,R.G.: "An optimal routing algorithm using distributed computation," IEEE Trans. on Communications, Vol.COM-25, pp.73-85 (1977).

[HASS 81] Hassin,R.: "Maximum flow in (s,t) planar networks, Inform. Proc. Lett., Vol.13, pp.107 (1981).

[HOFF 56] Hoffman,A.J., Kruskal,J.B.: "Integral boundary points of convex polyhedra", Annals of Mathematical Studies 38, Princeton University Press, pp.223-246 (1956).

[HU 69] Hu,T.C.: Integer Programming and Network Flows, Addison Wesley, Reading, Mass. (1969).

[IBAR 82] Ibaraki,T., Hosono,M. and Hasegawa,T.: "Network flow approaches to multi-item multi-stage production scheduling", Working paper, Dept. of Applied Mathmatics and Physics, Kyoto University, Kyoto, Oct. (1982).

[IBAR 76a] Ibaraki,T.: "Theoretical comparisons of search strategies in branch-and-bound algorithms", International J. Computer and Information Sciences, Vol.5, pp.315-344 (1976).

[IBAR 76b] Ibaraki,T.: "Computational efficiency of approximate branch-and-bound algorithms", Mathematics of Operations Res., Vol.1, pp.287-298 (1976).

[IBAR 77] Ibaraki,T.: "On the computational efficiency of branch-and-bound algorithms", Journal of Operations Research Society of Japan, Vol.20, pp.16-35 (1977).

[IBAR 78] Ibaraki,T.: "Branch-and-bound procedure and state-space representation of combinatorial optimization problems", Information and Control, Vol.36, pp.1-27 (1978).

[ITO 85] Ito,H.: "Performance evaluation of the algorithm for some multicommodity flow

problem" (in Japanese), Bachelor thesis, Dept. of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University (1985).

[IRI 70] Iri,M.: "On an extension of the maximum-flow minimum-cut theorem to multicommodity flows, Journal of the Operation Research Society of Japan, Vol.13, pp.129-135 (1970).

[KARM 84] Karmarkar,N.: "A new polynomial-time algorithm for linear programming", Combinatorica, Vol.4, No.4, pp.373-395 (1984).

[KARZ 85] Karzanov,A.V.: "Determining the maximal flow in a network by the method of preflows", Soviet Math. Dokl., Vol.15, pp.434-437 (1974).

[KENN 78] Kennington,J.L.: "A survey of linear cost multicommodity network flows", Operations Research, Vol.26, pp.209-236 (1978).

[KENN 80] Kennington,J.L. and Helgason,R.V.: Algorithms for Network Programing, John Wiley and Sons. (1980).

[KHAC 79] Khachiyan,L.G.: "A polynomial algorithm in linear programming", Dokl. Akad. Nouk SSSR N.S.

244:5, pp.1093-1096 (1979) [English transl., Soviet Math. Dokl. 20:1, pp.191-194 (1979)].

[LAWL 76] Lawler,E.L.: Combinatorial Optimization: Networks and Matroids, Holt, Rinehart & Winston, New York (1976).

[LOMO 85] Lomonosov,M.V.: Combinatorial Approaches to Multiflow Problems, Disc. Appl. Math., Vol.11, pp.1-93 (1985)

[MTSU 85] Matsumoto,K., Nishizeki,T. and Saito,N.: "An efficient algorithms for finding multicommodity flows in planar graphs", SIAM J. Comput., Vol.14, No.2, pp.289-320 (1985).

[NAGA 87a] Nagamochi,H., Ibaraki,T. and Hasegawa,T.: "On multicommodity flow problems in certain directed planar networks", Transactions of the Institute of Electronics, Information and Communication Engineers of Japan, Section A, pp.228-238 (1987).

[NAGA 87b] Nagamochi,H., Ibaraki,T. and Hasegawa,T.: "Multicommodity flow for the class CU of certain planar directed networks", Transactions of the Institute of Electronics, Information and Communication Engineers of Japan, Section A,

pp.1328-1339 (1987).

[NAGA 87c] Nagamochi,H., Fukushima,M and Ibaraki,T.: "Relaxation methods for the strictly convex multicommodity flow problem with capacity constraints on individual commodities", Journal of the Operation Research Society of Japan (submitted).

[NAGA 88a] Nagamochi,H. and Ibaraki,T.: "Max-flow min-cut theorem for the multicommodity flows in certain directed planar networks", Transactions of the Institute of Electronics, Information and Communication Engineers of Japan, Section A, pp.71-82 (1988).

[NAGA 88b] Nagamochi,H. and Ibaraki,T.: "An efficient feasibility testing of the multicommodity flow problem in certain planar directed networks", Transactions of the Institute of Electronics, Information and Communication Engineers of Japan, Section A (to appear).

[NAGA 88c] Nagamochi,H. and Ibaraki,T.: "Multicommodity flows in certain planar directed networks", Disc. Appl. Math. (to appear).

[NEMH 72] Nemhauser,G. and Garfinkel,R.: Integer Programming, John Wiley, New York (1972).

[OKAM 81] Okamura,H. and Seymour,P.D.: "Multicommodity flows in planar graphs", Journal of Combinatorial Theory, Series B, Vol.31, pp.75-81 (1981).

[OKAM 83] Okamura,H.: "Multicommodity flows in graphs", Disc. Appl. Math., Vol.6, pp.55-62 (1983).

[ONAG 71] Onaga,K. and Kakusho,O.: "On feasibility conditions of multicommodity flows in networks", IEEE Trans. Circuit Theory Vol.18, pp.425-429 (1971).

[ORLI 84] Orlin,J.B.: "Genuinely polynomial simplex and non-simplex algorithms for the minimum cost flow problem", Working papar No.1615-84, A.P.Sloan School of Management, MIT, December (1984).

[ORLI 86] Orlin,J.B.: "A dual version of Tardos's algorism for linear programming", Operations Research Letters, Vol.5, No.5 (1986).

[PAPA 82] Papadimitriou,C. and Steiglitz,K.: Combinatorial Optimization: Algorithms and Complexity, Printice-Hall, Englewood Clifs, New

Jersey (1982).

[ROCK 70] Rockafellar,R.T.: Convex Analysis, Princeton Univ. Press, N.J. (1970).

[ROCK 81] Rockafellar,R.T.: Monotropic programming: descent algorithms and duality : in Nonlinear Programming 4, by Mangasarian,O.L., Meyer,R. and Robinson,S. (eds.), Academic Press, pp.327-366 (1981).

[ROCK 84] Rockarfellar,R.T.: Network Flows and Monotropic Optimization: Pure and Applied Mathematics, A Wiley-Intersicience Series of Texts, Monographs and Tracts (1984).

[SLEA 80] Sleator,D.D.: "An O(mnlogn) algorithm for maximam network flow", Ph.D. Dissertation, Computer Science Department, Stanford University (1980).

[STER 77] Stern,T.E.: "A class of descentralized routing algorithms using relaxation", IEEE Trans. on Communications, Vol.COM-25, pp.1092-1102 (1977).

[SUZU 85] Suzuki,H., Nishizeki,T. and Saito,N.: "Multicommodity flows in planar undirected

graphs and shortest paths", Proc. 17-th Annual ACM Symposium on Theory of Computing, pp.195-204 (1985).

[TANG 64] Tang,D.T.: "Bi-path networks and multicommodity flows", IEEE Trans. Circuit Theory, Vol.CT-11, pp.468-474 (1964).

[TARD 86] Tardos,E.: "A strongly polynomial algorithm to solve combinatorial linear programs", Operations Resaerch, Vol.34, No.2, March-April, pp.250-256 (1986).

[TARD 85] Tardos,E.: "A strongly polynomial minimum cost circulation algorithm", Combinatorica, Vol.5, No.3, pp.247-255 (1985).

[TARJ 83] Tarjan,R.E.: Data Structures and Network Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, Penn. (1983).

[ZAHO 84] Zahorik,A., Thomas,L.J. and Trigeiro,W.W.: "Network programming models for production scheduling in multi-stage, multi-item capacitated systems", Management Science, Vol.30, No.3, pp.308-325 (1984).