Flows in Networks

by

L. R. FORD, Jr., and D. R. FULKERSON

[illegible] 3 22259 [illegible]

E704

2175

Flows in Networks

BY

L. R. FORD, JR.

C–E–I–R, INC.

D. R. FULKERSON

THE RAND CORPORATION

1962

PRINCETON UNIVERSITY PRESS

PRINCETON, NEW JERSEY

Printed in the United States of America

PREFACE

This book presents one approach to that part of linear programming theory that has come to be encompassed by the phrase "transportation problems" or "network flow problems." We use the latter name, not only because it is more nearly suggestive of the mathematical content of the subject, but also because it is less committed to one domain of application. Since many of the applications that are examined have little to do with transportation (and we have not included all the different ways in which the theory has already been used), it seems appropriate not to stress one particular applied area over others.

Just where the study of network flow problems may be said to have originated is a debatable question. Certain static minimal cost transportation models were independently studied by Hitchcock, Kantorovitch, and Koopmans in the 1940's. A few years later, when linear programming began to make itself known as an organized discipline, Dantzig showed how his general algorithm for solving linear programs, the simplex method, could be simplified and made more effective for the special case of transportation models. It would not be inaccurate to say that the subject matter of this book began with the work of these men on the very practical problem of transporting a commodity from certain points of supply to other points of demand in a way to minimize shipping cost. (This problem forms the nucleus of our Chapter III, entitled "Minimal Cost Flow Problems.") However, dismissing the formulational and applied aspects of the subject completely, and with the advantages of hindsight, one can go back a few years earlier to research of König, Egerváry, and Menger on linear graphs, or Hall on systems of distinct representatives for sets, and also relate this work in pure mathematics to the practically oriented subject of flows in networks. We have done this in Chapter II, "Feasibility Theorems and Combinatorial Applications."

One characteristic of the book that has been suggested above should perhaps be made explicit. While this is primarily a book in applied mathematics, we have also included topics that are purely mathematically motivated, together with those that are strictly utilitarian in concept. For this, no apology is intended. We have simply written about mathematics which has interested us, pure or applied.

To carry the historical sketch another (and our last) step back in time might lead one to the Maxwell-Kirchhoff theory of current distribution in an electrical network. Although this topic is closely related to the subject of the book, we have chosen not to include it. The reason for this is that we have limited the flow problems discussed to purely linear ones and, within this category, to those for which the assumption of integral data in the problem implies the existence of an integral solution. This sub-class of linear flow problems has, we feel, a simple elegance not shared by those outside the class. The first restriction, that of linearity, eliminates the Maxwell-Kirchhoff electrical network problem, which, viewed as a programming problem, becomes one of minimizing a quadratic function subject to linear constraints. The second restriction eliminates, for example, linear problems that involve the simultaneous flow of several commodities, important as these may be in practical applications of linear programming.

There are four chapters in the pages that follow; two of them (Chapters II and III) have been mentioned already. Chapter I, "Static Maximal Flow," studies the problem of maximizing flow from one point to another in a capacity-constrained network. From our point of view, this problem is the most fundamental topic dealt with in the book. Its solution provides a method of attack on the feasibility and combinatorial questions that form the subject of Chapter II, while the simple construction that results, when taken in conjunction with work of Kuhn on the optimal assignment problem, provides the key to the development of the various minimal cost flow methods in Chapter III. In addition, the recent treatment by Gomory and Hu of multi-terminal maximal flows, which is presented in Chapter IV, relies heavily on the central theorem of Chapter I. Thus Chapter I is prerequisite to the others, which are largely independent of each other.

Throughout the book the emphasis is on constructive procedures, even more, on computationally effective ones. Other things being nearly equal, we prefer a constructive proof of a theorem to a non-constructive one, and a constructive proof that leads to an efficient computational scheme is, to our way of thinking, just that much better.

The reader who is familiar with the simplex method of solution for network flow problems will find that this facet of the subject has been omitted in our presentation. For example, the notion of a spanning subtree of a network, which would play a fundamental role in the simplex theory, is not introduced until the last chapter, and then for another use. This omission does not reflect an aesthetic judgment on our part; it is, rather, that the more purely combinatorial methods developed here seem to be better computationally and also yield fresh insight into the subject.

ACKNOWLEDGMENTS

It is a pleasure to record our obligation to the following people, with whom we have discussed various parts of the manuscript: S. E. Dreyfus, R. E. Gomory, T. C. Hu, H. J. Ryser, L. S. Shapley, P. Wolfe, and J. W. T. Youngs. We want also to express special appreciation to T. E. Harris and F. S. Ross, who stimulated our interest in the topic of flows in networks, and to G. B. Dantzig, who has provided steady encouragement in our work on this topic over the past several years.

This study was prepared as part of the continuing program of research undertaken for the U.S. Air Force by The RAND Corporation.

CONTENTS

CHAPTER I

STATIC MAXIMAL FLOW

Introduction

The mathematical problem which forms the subject matter of this chapter, that of determining a maximal steady state flow from one point to another in a network subject to capacity limitations on arcs, comes up naturally in the study of transportation or communication networks. It was posed to the authors in the spring of 1955 by T. E. Harris, who, in conjunction with General F. S. Ross (Ret.), had formulated a simplified model of railway traffic flow, and pinpointed this particular problem as the central one suggested by the model [11]. It was not long after this until the main result, Theorem 5.1, which we call the max-flow min-cut theorem, was conjectured and established [4]. A number of proofs of this theorem have since appeared [2, 3, 5, 14]. The constructive proof given in § 5 is the simplest and most revealing of the several known to us.

Sections 1 and 2 discuss networks and flows in networks. There are many alternative ways of formulating the concept of a flow through a network; two of these are described in § 2. After introducing some notation in § 3, and defining the notion of a cut in § 4, we proceed to a statement and proof of the max-flow min-cut theorem in § 5. Sections 6, 7, 9, 10, and 11 amplify and extend this theorem. In § 8, the construction implicit in its proof is detailed and illustrated. This construction, which we call the "labeling process," forms the basis for almost all the algorithms presented later in the book. A consequence of the construction is the integrity theorem (Theorem 8.1), which has been known in connection with similar problems since G. B. Dantzig pointed it out in 1951 [1]. It is this theorem that makes network flow theory applicable in certain combinatorial investigations.

Section 12 provides a brief presentation of duality principles for linear programs. Since no proofs are included, the reader not familiar with linear inequality theory may find this section too brief to be very illuminating. But excellent discussions are available [8, 9, 10]. We include § 12 mainly to note that the max-flow min-cut theorem is a kind of combinatorial counterpart, for the special case of the maximal flow problem, of the more general duality theorem for linear programs.

Section 13 uses the max-flow min-cut theorem to examine maximal flow through a network as a function of a pair of individual arc capacities. The main conclusion here, which may sound empty but is not, is that any two arcs either always reinforce each other or always interfere with each other.

1. Networks

A *directed network* or *directed linear graph* $G = [N; \mathscr{A}]$ consists of a collection N of elements $x, y, \ldots$, together with a subset $\mathscr{A}$ of the ordered pairs (x, y) of elements taken from N. It is assumed throughout that N is a finite set, since our interest lies mainly in the construction of computational procedures. The elements of N are variously called *nodes, vertices, junction points,* or *points*; members of $\mathscr{A}$ are referred to as *arcs, links, branches,* or *edges*. We shall use the node-arc terminology throughout.

A network may be pictured by selecting a point corresponding to each node x of N and directing an arrow from x to y if the ordered pair (x, y) is in $\mathscr{A}$. For example, the network shown in Fig. 1.1 consists of four nodes s, x, y, t, and six arcs (s, x), (s, y), (x, y), (y, x), (x, t) and (y, t).

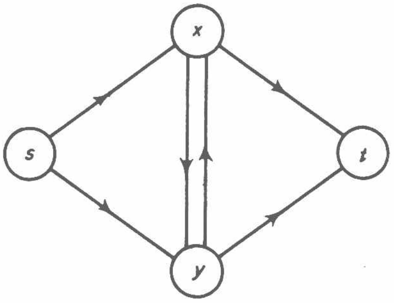

Figure 1.1

Such a network is said to be directed, since each arc carries a specific orientation or direction. Occasionally we shall also consider *undirected networks*, for which the set $\mathscr{A}$ consists of unordered pairs of nodes, or *mixed networks*, in which some arcs are directed, others are not. We can of course picture these in the same way, omitting arrowheads on arcs having no orientation. Until something is said to the contrary, however, each arc of the network will be assumed to have an orientation.

We have not as yet ruled out the possibility of arcs (x, x) leading from a node x to itself, but for our purposes we may as well do so. Thus, all arcs will be supposed to be of the form (x, y) with $x \neq y$. Also, while the existence of at most one arc (x, y) has been postulated, the notion of a network frequently allows multiple arcs joining x to y. Again, for most of the problems we shall consider, this added generality gains nothing, and so we shall continue to think of at most one arc leading from any node to another, unless an explicit statement is made to the contrary.

Let $x_1, x_2, \ldots, x_n$ $(n \geqslant 2)$ be a sequence of distinct nodes of a network such that (x_i, x_{i+1}) is an arc, for each $i = 1, \ldots, n - 1$. Then the sequence of nodes and arcs

$$x_1, (x_1, x_2), x_2, \ldots, (x_{n-1}, x_n), x_n \tag{1.1}$$

is called a *chain*; it leads from x_1 to x_n. Sometimes, for emphasis, we call (1.1) a *directed chain*. If the definition of a chain is altered by stipulating that $x_n = x_1$, then the displayed sequence is a *directed cycle*. For example, in the network of Fig. 1.1, the chain s, (s, x), x, (x, t), t leads from s to t; this network contains just one directed cycle, namely, x, (x, y), y, (y, x), x.

Let $x_1, x_2, \ldots, x_n$ be a sequence of distinct nodes having the property that either (x_i, x_{i+1}) or (x_{i+1}, x_i) is an arc, for each $i = 1, \ldots, n - 1$. Singling out, for each i, one of these two possibilities, we call the resulting sequence of nodes and arcs a *path from x_1 to x_n*. Thus a path differs from a chain by allowing the possibility of traversing an arc in a direction opposite to its orientation in going from x_1 to x_n. (For undirected networks, the two notions coincide.) Arcs (x_i, x_{i+1}) that belong to the path are *forward* arcs of the path; the others are *reverse* arcs. For example, the sequence s, (s, y), y, (x, y), x, (x, t), t is a path from s to t in Fig. 1.1; it contains the forward arcs (s, y), (x, t) and the reverse arc (x, y). If, in the definition of path, we stipulate that $x_1 = x_n$, then the resulting sequence of nodes and arcs is a *cycle*.

We may shorten the notation and refer unambiguously to the chain $x_1, x_2, \ldots, x_n$. Occasionally we shall also refer to the path $x_1, x_2, \ldots, x_n$; then it is to be understood that some selection of arcs has tacitly been made.

Given a network $[N; \mathscr{A}]$, one can form a *node-arc incidence matrix* as follows. List the nodes of the network vertically, say, the arcs horizontally, and record, in the column corresponding to arc (x, y), a 1 in the row corresponding to node x, a -1 in the row corresponding to y, and zeros elsewhere. For example, the network of Fig. 1.1 has incidence matrix

$$\begin{array}{c} \\ s \\ x \\ y \\ t \end{array}\begin{array}{c} \begin{array}{cccccc} (s,x) & (s,y) & (x,y) & (y,x) & (x,t) & (y,t) \end{array} \\ \left[\begin{array}{cccccc} 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 & 1 & 0 \\ 0 & -1 & -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 \end{array}\right] \end{array}.$$

Clearly, all information about the structure of a network is embodied in its node-arc incidence matrix.

If $x \in N$, we let $A(x)$ ("after x") denote the set of all $y \in N$ such that $(x, y) \in \mathscr{A}$:

$$A(x) = \{y \in N \mid (x, y) \in \mathscr{A}\}. \tag{1.2}$$

Similarly, we let $B(x)$ ("before x") denote the set of all $y \in N$ such that $(y, x) \in \mathscr{A}$:

$$B(x) = \{y \in N | (y, x) \in \mathscr{A}\}. \tag{1.3}$$

For example, in the network of Fig. 1.1,

$$A(s) = \{x, y\}, \qquad B(s) = \varnothing \text{ (the empty set).}$$

We shall on occasion require some other notions concerning networks. These will be introduced as needed.

2. Flows in networks

Given a network $G = [N; \mathscr{A}]$, suppose that each arc $(x, y) \in \mathscr{A}$ has associated with it a non-negative real number $c(x, y)$. We call $c(x, y)$ the *capacity* of the arc (x, y); it may be thought of intuitively as representing the maximal amount of some commodity that can arrive at y from x per unit time. The function c from $\mathscr{A}$ to non-negative reals is the *capacity function.* (Sometimes it will be convenient to allow infinite arc capacities also.)

The fundamental notion underlying most of the topics treated subsequently is that of a static or steady state flow through a network, which we now proceed to formulate. (Since dynamic flows will not be discussed until Chapter III, the qualifying phrase "static" or "steady state" will usually be omitted.)

Let s and t be two distinguished nodes of N. A *static flow of value v from s to t* in $[N; \mathscr{A}]$ is a function f from $\mathscr{A}$ to non-negative reals that satisfies the linear equations and inequalities

$$\sum_{y \in A(x)} f(x, y) - \sum_{y \in B(x)} f(y, x) = \begin{cases} v, & x = s, \\ 0, & x \neq s, t, \\ -v, & x = t, \end{cases} \tag{2.1}$$

$$f(x, y) \leqslant c(x, y) \qquad \text{all } (x, y) \in \mathscr{A}. \tag{2.2}$$

We call s the *source*, t the *sink*, and other nodes *intermediate*. Thus if the *net flow out of x* is defined to be

$$\sum_{y \in A(x)} f(x, y) - \sum_{y \in B(x)} f(y, x),$$

then the equations (2.1) may be verbalized by saying that the net flow out of the source is v, the net flow out of the sink is $-v$ (or the net flow into the sink is v), whereas the net flow out of an intermediate node is zero. An equation of the latter kind will be called a *conservation equation.*

When necessary to avoid ambiguity, we shall denote the value of a flow f by $v(f)$. Notice that a flow f from s to t of value v is a flow from t to s of value $-v$.

An example of a flow from s to t is shown in Fig. 2.1, where it is assumed that arc capacities are sufficiently large so that none are violated. The value of this flow is 3.

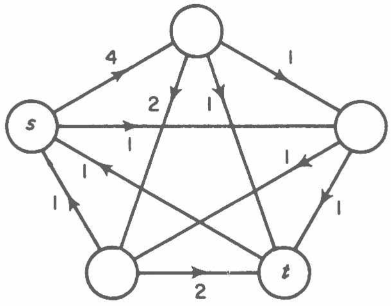

Figure 2.1

Given a flow f, we refer to $f(x, y)$ as the *arc flow* $f(x, y)$ or the *flow in arc* (x, y). Each arc flow $f(x, y)$ occurs in precisely two equations of (2.1), and has a coefficient 1 in the equation corresponding to node x, a coefficient -1 in the equation corresponding to node y. In other words, the coefficient matrix of equations (2.1), apart from the column corresponding to v, is the node-arc incidence matrix of the network. (By adding the special arc (t, s) to the network, allowing multiple arcs if necessary, a non-negative flow value v can be thought of as the "return flow" in (t, s), and all equations taken as conservation equations.)

A few observations. There is no question concerning the existence of flows, since $f = 0$, $v = 0$ satisfy (2.1) and (2.2). Also, while we have assumed that $\mathscr{A}$ may be a subset of the ordered pairs (x, y), $x \neq y$, with the capacity function c non-negative on $\mathscr{A}$, we could extend $\mathscr{A}$ to all ordered pairs by taking $c = 0$ outside of $\mathscr{A}$, or we could assume strict positivity of c by deleting from $\mathscr{A}$ arcs having zero capacity. Finally, the set of equations (2.1) is redundant, since adding the rows of its coefficient matrix produces the zero vector. Thus we could omit any one of the equations without loss of generality. We prefer, however, to retain the one-one correspondence between equations and nodes.

The static maximal flow problem is that of maximizing the variable v subject to the flow constraints (2.1) and (2.2). Before proceeding to this problem, it is worth while to point out an alternative formulation that is informative and will be useful in later contexts. This might be termed the arc-chain notion of a flow from s to t.

Suppose that $A_1, \ldots, A_m$ is an enumeration of the arcs of a network, the arc A_i having capacity $c(A_i)$; and let $C_1, \ldots, C_n$ be a list of all directed chains from s to t. Form the m by n incidence matrix (a_{ij}) of arcs versus chains by defining

$$a_{ij} = \begin{cases} 1, & \text{if } A_i \in C_j, \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

Now let h be a function from the set of chains $C_1, \ldots, C_n$ to non-negative reals that satisfies the inequalities

$$\sum_{j=1}^{n} a_{ij} h(C_j) \leqslant c(A_i), \qquad i = 1, \ldots, m. \tag{2.4}$$

We refer to h as a *flow from s to t in arc-chain form*, and call $h(C_j)$ a *chain flow* or the *flow in chain C_j*. The *value* of h is

$$v(h) = \sum_{j=1}^{n} h(C_j). \tag{2.5}$$

When we need to distinguish the two notions of a flow from s to t thus far introduced, we shall call a function f from the set of arcs to non-negative reals which satisfies (2.1) and (2.2) for some v, a *flow from s to t in node-arc form*. There will usually be no need for the distinction, since we shall work almost exclusively with node-arc flows after this section.

Let us explore the relationship between these two formulations of the intuitive notion of a flow. Suppose that $x_1, \ldots, x_l$ is a list of the nodes, and let (b_{ki}), $k = 1, \ldots, l$, $i = 1, \ldots, m$, be the node-arc incidence matrix introduced earlier. Thus

$$b_{ki} = \begin{cases} 1, & \text{if } A_i = (x_k, y), \\ -1, & \text{if } A_i = (y, x_k), \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

Then

$$b_{ki} a_{ij} = \begin{cases} 1, & \text{if } A_i = (x_k, y) \text{ and } A_i \in C_j, \\ -1, & \text{if } A_i = (y, x_k) \text{ and } A_i \in C_j, \\ 0, & \text{otherwise,} \end{cases}$$

and it follows that

$$\sum_{i=1}^{m} b_{ki} a_{ij} = \begin{cases} 1, & \text{if } x_k = s, \\ -1, & \text{if } x_k = t, \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

If h is a flow from s to t in arc-chain form, and if we define

$$f(A_i) = \sum_{j=1}^{n} a_{ij} h(C_j), \qquad i = 1, \ldots, m, \tag{2.8}$$

then f is a flow from s to t in node-arc form, and $v(f) = v(h)$. For, by (2.4) and (2.8),

$$f(A_i) \leqslant c(A_i),$$

and by (2.7),

$$\begin{aligned}\sum_{i=1}^{m} b_{ki}f(A_i) &= \sum_{i=1}^{m}\sum_{j=1}^{n} b_{ki}a_{ij}h(C_j)\\ &= \sum_{j=1}^{n}\left(\sum_{i=1}^{m} b_{ki}a_{ij}\right)h(C_j)\\ &= \begin{cases}\displaystyle\sum_{j=1}^{n} h(C_j), & \text{if } x_k = s,\\ \displaystyle -\sum_{j=1}^{n} h(C_j), & \text{if } x_k = t,\\ 0, & \text{otherwise.}\end{cases}\end{aligned}$$

But these are precisely equations (2.1) for the function f and $v = \sum_{j=1}^{n} h(C_j)$.

On the other hand, we can start with a flow f in node-arc form having value v, and obtain from it a flow h in arc-chain form having value $v(h) \geqslant v$. Intuitively, the reason the inequality now appears is that the node-arc formulation permits flow along chains from t to s.

There are various ways of obtaining such an arc-chain flow h from a given node-arc flow f. One way is as follows. Define

$$h(C_j) = \min_{A_i \in C_j} f_j(A_i), \qquad j = 1, \ldots, n, \tag{2.9}$$

where

$$f_j(A_i) = f(A_i) - \sum_{p=1}^{j-1} a_{ip}h(C_p), \qquad j = 1, \ldots, n+1. \tag{2.10}$$

In words, look at the first chain C_1, reduce $f_1 = f$ by as much as possible (retaining non-negativity of arc flows) on arcs of C_1; this yields f_2. The process is then repeated with C_2 and f_2, and so on until all chains have been examined. It follows that f_{j+1} is a node-arc flow from s to t having value $v(f_{j+1}) = v - \sum_{p=1}^{j} h(C_p)$, since

$$\begin{aligned}\sum_{i=1}^{m} b_{ki}f_{j+1}(A_i) &= \sum_{i=1}^{m} b_{ki}f(A_i) - \sum_{i=1}^{m}\sum_{p=1}^{j} b_{ki}a_{ip}h(C_p),\\ &= \begin{cases} v - \displaystyle\sum_{p=1}^{j} h(C_p), & \text{if } x_k = s,\\ -v + \displaystyle\sum_{p=1}^{j} h(C_p), & \text{if } x_k = t,\\ 0, & \text{otherwise.}\end{cases}\end{aligned}$$

Moreover, $f_{j+1}(A_i) \leqslant f_j(A_i)$, all A_i, and $f_{j+1}(A_i) = 0$ for some $A_i \in C_j$. Hence the node-arc flow f_{n+1} vanishes on some arc of every chain from s to t. This implies that $v(f_{n+1}) \leqslant 0$, as the following lemma shows.

Lemma 2.1. *If f is a node-arc flow from s to t having value $v(f) > 0$, then there is a chain from s to t such that $f > 0$ on all arcs of this chain.*

Proof. Let X be the set of nodes defined recursively by the rules

(a) $s \in X$,

(b) if $x \in X$, and if $f(x, y) > 0$, then $y \in X$.

We assert that $t \in X$. For, suppose not. Then, summing the equations (2.1) over $x \in X$, and noting cancellations, we have

$$v(f) = \sum_{\substack{x \in X \\ y \notin X}} [f(x, y) - f(y, x)].$$

But by (b), if (x, y) is an arc with $x \in X$, $y \notin X$, then $f(x, y) = 0$. This and the last displayed equation contradict $v(f) > 0$. Thus $t \in X$. But for any $x \in X$, the definition of X shows that there is a chain from s to x such that $f > 0$ on arcs of this chain. Hence there is a chain from s to t with this property.

It follows from the lemma that the value of f_{n+1} is non-positive, that is

$$v(f_{n+1}) = v - \sum_{p=1}^{n} h(C_p) \leqslant 0.$$

Consequently $v(h) \geqslant v$. This proves

Theorem 2.2. *If h is an arc-chain flow from s to t, then f defined by* (2.8) *is a node-arc flow from s to t and $v(f) = v(h)$. On the other hand, if f is a node-arc flow from s to t, then h defined by* (2.9) *and* (2.10) *is an arc-chain flow from s to t, and $v(h) \geqslant v(f)$.*

A consequence of Theorem 2.2 is that it is immaterial whether the maximal flow problem is formulated in terms of the node-arc incidence matrix or the arc-chain incidence matrix. Thus, for example, since arcs of the form (x, s) or (t, x) can be deleted from $\mathscr{A}$ without changing the list of chains from s to t, we may always suppose in either formulation of the maximal flow problem that all source arcs point out from the source, and all sink arcs point into the sink. (For such networks, one has $v(h) = v(f)$ in the second part of Theorem 2.2 as well as the first part.)

A function h defined from f as in (2.9) and (2.10) will be termed a *chain decomposition* of f. A chain decomposition of f will, in general, depend on the ordering of the chains. For example, if in Fig. 1.1 we take $f = 1$ on all arcs, and take $C_1 = (s, x, t)$, $C_2 = (s, y, t)$, $C_3 = (s, x, y, t)$, $C_4 = (s, y, x, t)$, then $h(C_1) = h(C_2) = 1$, $h(C_3) = h(C_4) = 0$. But, examining the chains in reverse order would lead to $h(C_4) = h(C_3) = 1$, $h(C_2) = h(C_1) = 0$.

From the computational point of view, one would certainly suppose the node-arc formulation of the maximal flow problem to be preferable for most networks, since the number of chains from s to t is likely to be large compared to the number of nodes or the number of arcs. A computing procedure that required as a first step the enumeration of all chains from s to t would be of little value. There are less obvious reasons why the node-arc formulation is to be preferred from the theoretical point of view as well.†

3. Notation

To simplify the notation, we adopt the following conventions. If X and Y are subsets of N, let (X, Y) denote the set of all arcs that lead from $x \in X$ to $y \in Y$; and, for any function g from $\mathscr{A}$ to reals, let

$$\sum_{(x,y)\in(X,Y)} g(x, y) = g(X, Y). \tag{3.1}$$

Similarly, when dealing with a function h defined on the nodes of N, we put

$$\sum_{x \in X} h(x) = h(X). \tag{3.2}$$

We customarily denote a set consisting of one element by its single element. Thus if X contains the single node x, we write (x, Y), $g(x, Y)$, and so on.

Set unions, intersections, and differences will be denoted by $\cup$, $\cap$, and $-$, respectively. Thus $X \cup Y$ is the set of nodes in X or in Y, $X \cap Y$ the set of nodes in both X and Y, and $X - Y$ the set of nodes in X but not in Y. We use $\subseteq$ for set inclusion, and $\subset$ for proper inclusion. Complements of sets will be denoted by barring the appropriate symbol. For instance, the complement of X in N is $\bar{X} = N - X$.

Thus, if $X, Y, Z \subseteq N$, then

$$g(X, Y \cup Z) = g(X, Y) + g(X, Z) - g(X, Y \cap Z), \tag{3.3}$$

$$g(Y \cup Z, X) = g(Y, X) + g(Z, X) - g(Y \cap Z, X). \tag{3.4}$$

Hence if Y and Z are disjoint,

$$g(X, Y \cup Z) = g(X, Y) + g(X, Z),$$

$$g(Y \cup Z, X) = g(Y, X) + g(Z, X).$$

† Two comments are in order here. First, one can describe a computing procedure for the arc-chain formulation of the maximal flow problem that does not require an explicit enumeration of all chains [6]. Second, a strong theoretical reason for adopting the node-arc formulation, nonetheless, is that the node-arc incidence matrix has a desirable property not shared by the arc-chain incidence matrix. This is the unimodularity property, that is, every submatrix has determinant ± 1 or 0. See [12] for a full discussion of this property and its implications for linear programming problems.