Mathematical Programming 6 (1974) 1--13. North-Holland Publishing Company

PACKING ROOTED DIRECTED CUTS IN A WEIGHTED DIRECTED GRAPH*

D.R. FULKERSON
Cornell University, Ithaca, New York, U.S.A.

Received 20 February 1973

A simple algorithm is described for constructing a maximum packing of cuts directed away from a distinguished vertex, called the root, in a directed graph, each of whose edges has a nonnegative weight, and it is shown that the maximum packing value is equal to the weight of a minimum-weight spanning arborescence directed away from the root.

1. Introduction

Let G be a directed graph with vertex set N and directed edges denoted by ordered pairs of vertices, the edge (i, j) being *directed from* $i \in N$ *to* $j \in N$. Let $r \in N$ be a distinguished vertex of G, called the *root* of G. (For the problems we are going to consider, we could assume, without loss of generality, that all edges of G incident with r are directed from r.) By a *cut directed away from* r we mean the set of all edges of G that are directed from a subset $X \subseteq N$, where $r \in X$, to $N \setminus X = \bar{X}$, where $\bar{X} \neq \emptyset$. We denote such a cut by $(X, \bar{X})$ and call it, for short, *a rooted directed cut* or an *r-directed cut*. Suppose that each edge (i, j) of G has a nonnegative integer *weight* $w(i, j)$. By a *packing of r-directed cuts* in the weighted graph G we mean the following: Assign each r-directed cut $(X, \bar{X})$ a nonnegative weight $y_{\bar{X}}$ in such a way that the sum of all the weights of r-directed cuts that contain a particular edge (i, j) does not exceed the weight $w(i, j)$ of that edge. A *maximum packing of r-directed cuts* is a packing in which the sum of the weights assigned to r-directed cuts is as large as possible. The main problem we treat is that of constructing a maximum packing of rooted directed cuts.

* This work was supported by the National Science Foundation under grant GP-32316X and partially by the Office of Naval Research under grant N00014-67-A-0077-0028.

A *spanning arborescence in G rooted at r* is a spanning tree of the underlying undirected graph of G, having the properties: (i) each vertex of G other than r has just one edge of the arborescence directed toward it, (ii) no edge of the arborescence is directed toward r. A *branching* in G is a forest of the underlying undirected graph whose edges are directed toward different vertices. In [2], Edmonds has described an efficient algorithm for constructing a maximum-weight branching in a weighted graph, and has determined linear inequalities that define the convex hull of the (0, 1)-incidence vectors of all branchings in a directed graph. In [8], it was asserted without proof that the algorithm of [2] can be modified to solve the maximum packing problem for rooted directed cuts. We shall describe this modification in Section 3, where it will also be shown that the packing problem always has an integer solution vector y whose component sum is equal to the weight of a minimum-weight spanning arborescence rooted at r. In the context of blocking pairs of matrices or blocking pairs of polyhedra [6, 8], if we let matrix A be the (0, 1)-incidence matrix of all spanning arborescences of G rooted at r (rows of A) versus all directed edges of G (columns of A), and similarly, let C denote the (0, 1)-incidence matrix of all (set-wise) minimal r-rooted cuts of G versus edges of G, then the max–min equality holds strongly for the ordered pair (C, A). That is, the linear program

$$\begin{aligned} &yC \leq w, \qquad y \geq 0\,, \\ &\max 1 \cdot y, \qquad \text{where } 1 = (1, \ldots, 1)\,, \end{aligned} \tag{1.1}$$

has an integer solution vector y whenever w is a nonnegative integer vector, and this solution vector y satisfies

$$1 \cdot y = \min_{1 \leq i \leq m} a^i \cdot w\,, \tag{1.2}$$

where matrix A has rows $a^1, \ldots, a^m$.

The incidence matrices A and C constitute a blocking pair of matrices, that is, the unbounded convex polyhedra

$$\mathcal{C} = \{c \geq 0\colon Ac \geq 1\}\,, \tag{1.3}$$

$$\mathcal{A} = \{a \geq 0\colon Ca \geq 1\}\,, \tag{1.4}$$

are a blocking pair of polyhedra [6, 8]. The extreme points of $\mathcal{C}$ are the rows of matrix C and the extreme points of $\mathcal{A}$ are the rows of matrix A. (It may happen that A has no rows, i.e., the graph G has no

spanning arborescence rooted at r, in which case matrix C has just one row, all its entries being zero. In this case the maximum in (1.1) and the minimum in (1.2) are infinite. The algorithm of Section 3 will detect this situation, although there are easier methods to detect this case initially.)

Edmonds' results in [2] fall naturally in the domain of anti-blocking pairs of polyhedra [7, 8], rather than blocking pairs of polyhedra. Indeed, the anti-blocking polyhedron of the convex hull of all branchings is determined explicitly in [2]. Before describing a modification of Edmonds' algorithm for optimum branchings that solves the maximum packing program (1.1), we proceed in Section 2 to a proof that the polyhedra $\mathcal{A}$ and $\mathcal{C}$ defined by (1.4) and (1.3) are a blocking pair. This proof also uses results of [2].

2. The polyhedra $\mathcal{A}$ and $\mathcal{C}$

Let $x(i, j)$ be a real variable associated with the directed edge (i, j) of a directed graph G. If $X \subseteq N$, $Y \subseteq N$, we use the notation

$$x(X, Y) = \sum_{\substack{i \in X \\ j \in Y}} x(i, j) . \tag{2.1}$$

It is proved in [2] that the extreme points of the bounded polyhedron $\mathcal{B}$ defined by the linear inequalities

$$x(i, j) \geq 0, \qquad \text{for all edges } (i, j) \text{ of } G , \tag{2.2}$$

$$x(N, j) \leq 1, \qquad \text{for all } j \in N , \tag{2.3}$$

$$x(X, X) \leq |X|-1, \qquad \text{for all nonempty } X \subseteq N , \tag{2.4}$$

are precisely the incidence vectors of all branchings in G. (In (2.4), $|X|$ denotes the cardinality of X.)

Theorem 2.1. *The polyhedra $\mathcal{A}$ and $\mathcal{C}$ defined by* (1.4) *and* (1.3) *for an r-rooted directed graph G are a blocking pair.*

Proof. It suffices to show (see [6]) that the extreme points of $\mathcal{A}$ are the incidence vectors of spanning arborescences rooted at r, i.e., are the rows of matrix A.

Let a be the incidence vector of a spanning arborescence rooted at r.

Then clearly $a \in \mathcal{A}$. For each vertex $j \neq r$ of G, let $\bar{X}_j$ denote the subset of all vertices $t \in N$ such that the unique directed path from r to t of the arborescence contains the unique edge of the arborescence directed into j, and let $X_j = N \setminus \bar{X}_j$. (In other words, if (i, j) is the edge of the arborescence directed into $j \neq r$, and if we delete (i, j) from the arborescence, the resulting subgraph is a branching having two (weak) components, one having vertex set X_j, with $r \in X_j$, and the other having vertex set $\bar{X}_j = N \setminus X_j$.) Then vector a is the unique solution of the set of equations

$$a(i, j) = 0, \qquad \text{for all } (i, j) \text{ not in the arborescence}, \tag{2.5}$$

$$a(X_j, \bar{X}_j) = 1, \quad \text{for all } j \neq r. \tag{2.6}$$

Hence vector a is extreme in $\mathcal{A}$.

Let a be an extreme point of $\mathcal{A}$. Then $a(N, r) = 0$, for otherwise a would be the midpoint of a line segment joining two distinct points of $\mathcal{A}$. We show next that $a(N, t) = 1$ for each $t \neq r$. For suppose $a(N, t) > 1$ for some $t \neq r$. Let $a(u_1, t), \ldots, a(u_k, t)$ denote the positive members of the sum $a(N, t)$. Since a is extreme in $\mathcal{A}$, we claim there exist r-directed cuts $(X_1, \bar{X}_1), \ldots, (X_k, \bar{X}_k)$, such that $(u_i, t) \in (X_i, \bar{X}_i)$ for $i = 1, \ldots, k$ and such that $a(X_i, \bar{X}_i) = 1$ for $i = 1, \ldots, k$. For if this were not so, we could "wiggle" the component $a(u_i, t)$ of a by subtracting and adding some $\epsilon > 0$ to it, obtaining two points of $\mathcal{A}$ having midpoint a. Now view the vector a as a capacity vector on the flow network G with source r and sink t [5]. Each of the cuts $(X_i, \bar{X}_i)$ is a minimum-capacity cut separating r and t in this flow network, and it follows from [5, Corollary I.5.4] that the r-directed cut $(\bigcup X_i, \bigcap \bar{X}_i)$ is also a minimum-capacity cut separating r and t. Thus $a(\bigcup X_i, \bigcap \bar{X}_i) = 1$. Hence

$$1 < a(N, t) \leq a(\bigcup X_i, \bigcap \bar{X}_i) = 1,$$

a contradiction. Thus $a(N, t) = 1$ for each $t \neq r$.

The vector a thus satisfies (2.2) and (2.3). It also satisfies (2.4), since for $X \subseteq N$, $X \neq \emptyset$, we have, using $a(N, r) = 0$ and $a(N, t) = 1$ for $t \neq r$,

$$\begin{aligned} a(X, X) &= a(N, N) - a(N, \bar{X}) - a(\bar{X}, X), \\ &\leq |N| - 1 - (|N| - |X|) = |X| - 1. \end{aligned}$$

Thus $a \in \mathcal{B}$, and hence, by Edmonds' theorem [2, Theorem 2], a is a convex combination of the incidence vectors $b^1, \ldots, b^n$ of branchings. But since $a(N, r) = 0$ and $a(N, t) = 1$ for $t \neq r$, the same equations hold

for each branching $b^1, \ldots, b^n$. Thus each of these branchings is a spanning arborescence rooted at r, hence $b^1, \ldots, b^n$ are points of $\mathcal{A}$. Since a is extreme in $\mathcal{A}$, we must have $a = b^1 = \ldots = b^n$, and thus the vector a is the incidence vector of a spanning arborescence rooted at r. This completes the proof of Theorem 2.1.

It is a consequence of Theorem 2.1 and of results of [6] that the max–min equality holds for both ordered pairs (A, C) and (C, A). But it does not follow necessarily from these results that the max–min equality holds strongly in either case, even though both A and C are (0, 1)-matrices. In the next section we shall prove that the max–min equality does hold strongly for (C, A) by a modification of Edmonds' algorithm for optimum branchings [2]. Edmonds has subsequently proved the very interesting result that it also holds strongly for (A, C) [3, 4]. That is, the linear packing program

$$\begin{aligned} &yA \leq w, \qquad y \geq 0, \\ &\max 1 \cdot y, \end{aligned} \tag{2.7}$$

always has an integer solution vector y whenever w is a nonnegative integer vector, and this integer vector y has component sum equal to the weight of a minimum-weight r-directed cut,

$$1 \cdot y = \min_{1 \leq j \leq n} c^j \cdot w, \tag{2.8}$$

where matrix C has rows $c^1, \ldots, c^n$. Results of [6, 8] would imply only the existence of a rational vector y satisfying (2.7) and (2.8).

A further consequence of Theorem 2.1 and of [6] is that the min–min inequality holds for the matrices A and C. (This is the analogue of the length–width inequality for paths joining two terminals of a graph and cuts separating the terminals [1, 9].) The min–min inequality for A and C asserts the following. Let l and w be two nonnegative vectors, each having one component for each edge of an r-rooted directed graph. Then the inequality

$$\{\min_{1 \leq i \leq m} a^i \cdot l\} \; \{\min_{1 \leq j \leq n} c^j \cdot w\} \leq l \cdot w \tag{2.9}$$

always holds, where A has rows $a^1, \ldots, a^m$, and C has rows $c^1, \ldots, c^n$. In other words, the weight of a minimum-weight spanning arborescence rooted at r, computed using the weight vector l, times the weight of a

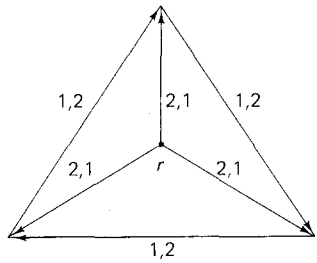


Fig. 2.1.

minimum-weight r-directed cut, computed using the weight vector w, is at most equal to the inner product of vectors l and w. (For an example of (2.9), see Fig. 2.1, where the components of l and w are recorded as first and second members, respectively, on the edges. Each side of (2.9) is equal to 12 in the example.) A direct proof of (2.9) would prove Theorem 2.1.

3. Algorithm and example.

The algorithm for constructing a maximum packing of r-directed cuts in a weighted graph is extremely simple. It has the very nice feature that nonnegative weights (components of y in (1.1)) are assigned to certain r-directed cuts sequentially, and that once such a weight has been assigned, that weight is never changed subsequently in the course of the algorithm. (Positive weights can be assigned in the algorithm to r-directed cuts that are not set-wise minimal, but this does not matter. Such weights could be transferrred to rows of matrix C if desired. We shall not bother to do so.)

The proof that the algorithm does construct a maximum packing of r-directed cuts is somewhat more involved. It requires showing that the algorithm can be continued (see I.3 of [2]) to produce a minimum-weight spanning arborescence rooted at r whose weight sum equals the packing sum. The complete algorithm for constructing both a maximum packing of r-directed cuts and a minimum-weight spanning arborescence rooted at r is a modification of that described in [2]. It is simpler, however, particularly in the assignment of (what may be regarded, from the point of view of constructing a minimum-weight spanning arborescence rooted at r) optimal dual variables, which here solve the packing problem (our primal problem), but which in [2] solve

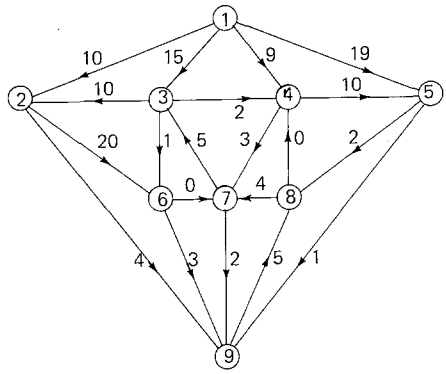


Fig. 3.1.

a certain covering problem (the linear programming dual of (2.2), (2.3) and (2.4)).

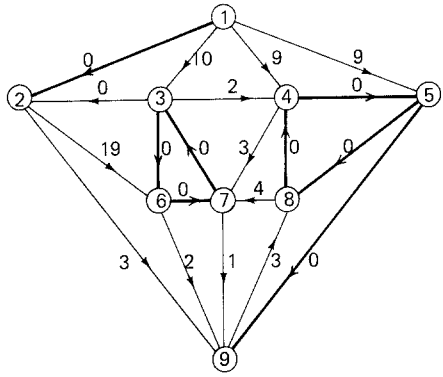
After describing the algorithm, we shall apply it to an illustrative example, the weighted graph of Fig. 3.1, with vertex 1 as the root.

Begin by selecting from each bundle of inwardly-directed edges $(N - t, t)$ at each $t \neq r$ one edge having least weight among all edges in $(N - t, t)$. For each vertex $t \neq r$, assign the r-directed cut $(N - t, t)$ the packing weight y_t equal to the weight of the edge selected from this cut, and reduce all weights of edges in this cut by y_t, obtaining a new non-negative weight vector w' on edges. If any one of the r-directed cuts $(N - t, t)$ for $t \neq r$ is empty, stop. (In this case, the maximum packing value is infinite and G has no spanning arborescence rooted at r.) Otherwise the subgraph of selected edges has $|N| - 1$ members; if this subgraph has p (weak) components, then it has precisely $p - 1$ directed circuits, one in each component not containing the root r. If $p = 1$, stop. (The present packing is maximum.) If $p > 1$, contract each of the $p - 1$ directed circuits to a new vertex in a new directed graph G', i.e., contract each edge of G that joins two vertices of one of these $p - 1$ directed circuits. Edges of G' have weights given by w'. (G' may of course have several edges directed from one vertex to another vertex, even if G did not. We can replace such multiple edges by one edge whose weight is equal to the least of these weights.)

The process is now repeated with G' and w', but in doing so we only look at the new vertices of G'.

Eventually, we find either an empty r-directed cut or $p = 1$.

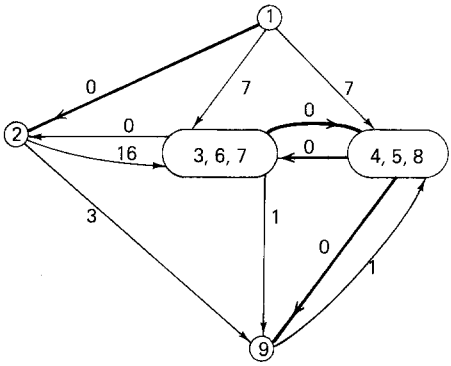
Figs. 3.2 – 3.5 show the construction for the example of Fig. 3.1. Selected edges are bold in the figures. Multiple edges have been replaced



G and $w'(p = 3)$.

y	Edge-list	w	w'
10	(1,2) ∨	10	0
	(3,2)	10	0
5	(1,3)	15	10
	(7,3) ∨	5	0
0	(1,4)	9	9
	(3,4)	2	2
	(8,4) ∨	0	0
10	(1,5)	19	9
	(4,5) ∨	10	0
1	(2,6)	20	19
	(3,6) ∨	1	0
0	(4,7)	3	3
	(6,7) ∨	0	0
	(8,7)	4	4
2	(5,8) ∨	2	0
	(9,8)	5	3
1	(2,9)	4	3
	(5,9) ∨	1	0
	(6,9)	3	2
	(7,9)	2	1

Fig. 3.2.

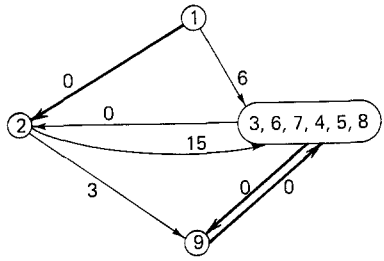


G' and $w''(p = 2)$.

y	Edge-list	w'	w''
3	(2,6)	19	16
	(4,7) ∨	3	0
	(1,3)	10	7
2	(1,4)	9	7
	(3,4) ∨	2	0
	(9,8)	3	1

Fig. 3.3.

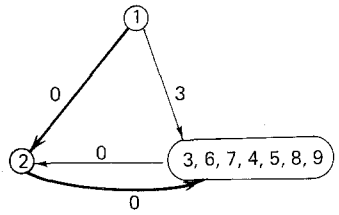
by one edge. Each new vertex of G' is identified by the set of vertex numbers of G that corresponded to the vertices of G contained in the directed circuit of G that was contracted to form the new vertex. The construction is shown in both edge-list form and in diagram form in the figures.



G'' and $w''''(p = 2)$.

y	Edge-list	w''	w''''
1	(1,3)	7	6
	(2,6)	16	15
	(9,8)∨	1	0

Fig. 3.4.



G''' and $w''''(p = 1)$.

y	Edge-list	w'''	w''''
3	(1,3)	6	3
	(2,9)∨	3	0

Fig. 3.5.

In Fig. 3.3, we have generated two new packing weights: 3 on the (nonempty) cut $(X, \overline{X})$ with $\overline{X} = \{3, 6, 7\}$ and 2 on the (nonempty) cut $(Y, \overline{Y})$ with $\overline{Y} = \{4, 5, 8\}$. Since $p = 2$, we contract the unique circuit of bold edges to obtain G'' shown in Fig. 3.4.

The only new packing weight obtained from Fig. 3.4 is 1 on the (nonempty) cut $(Z, \overline{Z})$, where $\overline{Z} = \{3, 6, 7, 4, 5, 8\}$. Again, $p = 2$ and we contract to obtain Fig. 3.5.

At this stage, $p = 1$ and we are done, having generated one new packing weight of 3 on the (nonempty) cut $(W, \overline{W})$, where $\overline{W} = \{3, 6, 7, 4, 5, 8, 9\}$. The total packing value is 38.

It is clear that the algorithm produces an integer packing of r-directed cuts. It remains to show that this packing is maximum. To accomplish this, one can work backward in the sequence of graphs produced by the algorithm (from Fig. 3.5 to Fig. 3.2 in the example) to produce a spanning arborescence rooted at r whose weight is equal to the packing value. The backward process retains certain selected edges and deletes others. The proof that it constructs a spanning arborescence rooted at r is like the proof given in [2] that the similar procedure there constructs a branching, the main difference being that we start with a spanning ar-

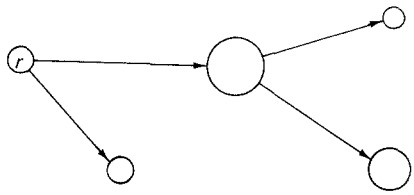


Fig. 3.6.

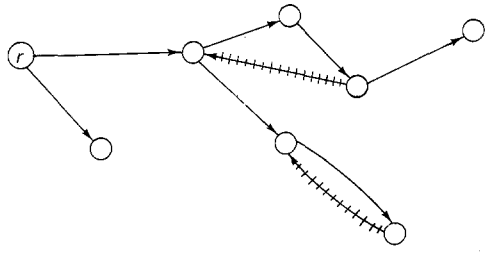


Fig. 3.7.

borescence rooted at r, and thus retain this property inductively. The idea of the inductive step is simply the following. Suppose we have a spanning arborescence rooted at r, with certain non-root vertices being distinguished (e.g., the "large" vertices of Fig. 3.6). If we replace each of these distinguished vertices by a directed circuit (e.g., as shown in Fig. 3.7) and then delete, for each of these circuits, the unique edge of the circuit that is directed toward the same vertex as one of the old edges, the new graph is again a spanning arborescence rooted at r.

Thus, in the example, we work backward in this manner from the spanning arborescence rooted at 1 of Fig. 3.5 as shown in Figs. 3.5′ – 3.2′. Fig. 3.2′ shows the spanning arborescence rooted at 1 of the original graph that is obtained together with the original weights on arborescence edges. The total weight is 38.

The backward replacement process can also be viewed globally, rather than sequentially, just in terms of the list of selected edges and the list of subsets $\bar{X}$ of the r-directed cuts $(X, \bar{X})$ that produced the selected edges, one for each cut. (See Fig. 3.8 for the example.)

Start with the last member of the edge-list, look for all preceding r-directed cuts containing this edge, and delete their corresponding edges in the edge-list. Repeat this procedure. (In Fig. 3.8, edge (2, 9) knocks out (5, 9), edge (9, 8) knocks out (3, 4) and (5, 8), edge (4, 7) knocks out (6, 7). The remaining edges form the spanning arborescence of Fig. 3.2′.)

An important point follows from this: Any r-directed cut $(X, \bar{X})$ produced in the packing has just one edge in common with the spanning arborescence rooted at r that is constructed. Thus

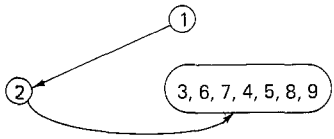


Fig. 3.5'.

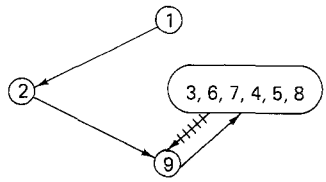


Fig. 3.4'.

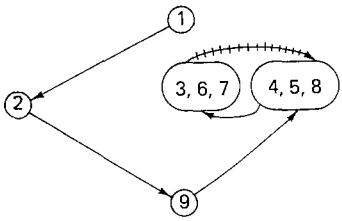


Fig. 3.3'.

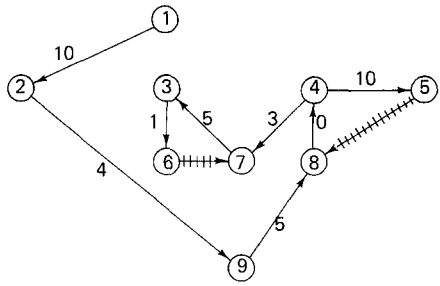


Fig. 3.2'.

$$y_{\bar{X}} > 0 \Rightarrow a \cdot c^{\bar{X}} = 1 \,, \tag{3.1}$$

where $y_{\bar{X}}$ is the packing weight assigned to $(X, \bar{X})$, $c^{\bar{X}}$ is the incidence vector of the cut $(X, \bar{X})$, and a is the incidence vector of the spanning arborescence rooted at r.

It is also clear that the sum of the weights $y_{\bar{X}}$ on cuts $(X, \bar{X})$ that contain a given edge of the arborescence is equal to the weight of that edge. Thus

Edges	Subsets $\bar{X} \subseteq N$
(1,2)	2
(7,3)	3
(8,4)	4
(4,5)	5
(3,6)	6
(6,7) V	7
(5,8) V	8
(5,9) V	9
(4,7)	3, 6, 7
(3,4) V	4, 5, 8
(9,8)	3, 6, 7, 4, 5, 8
(2,9)	3, 6, 7, 4, 5, 8, 9

Fig. 3.8.

$$a(i, j) > 0 \Rightarrow \sum_{(X, \bar{X}) \ni (i, j)} y_{\bar{X}} = w(i, j) . \tag{3.2}$$

Together, (3.1) and (3.2) show that the integer packing of r-directed cuts is maximum, the weight of the spanning arborescence rooted at r is minimum, and the packing value is equal to the arborescence weight. Thus

Theorem 3.1. *The max–min equality holds strongly for the ordered pair of incidence matrices* (C, A).

The algorithm picks out a square submatrix of the incidence matrix C' of r-directed cuts versus edges of G, which has one row for each cut selected in the algorithm and one column for each edge selected, and simultaneously arranges this submatrix in upper triangular form T, with 1's along the diagonal. T is thus nonsingular. In the first part of the algorithm, the equations $\bar{y}\, T = \bar{w}$, where $\bar{y}(\bar{w})$ denotes the vector of the appropriate components of $y(w)$ are solved, producing a nonnegative integer solution (and the inequalities $y\, C \leq w$, or $y\, C' \leq w$, are not violated). The second part of the algorithm solves the equations $T\bar{a} = 1$, and the resulting (0, 1)-solution vector $\bar{a}$ is the incidence vector of a spanning arborescence rooted at r. (See Fig. 3.9 for the example.)

Thus, in linear programming terminology, the cut packing program (1.1) always has an optimal basis which is triangular.

$\bar{y}$	$\bar{a}$		1	1	1	1	1	0	0	0	1	0	1	1	1
	Edges		(1,2)	(7,3)	(8,4)	(4,5)	(3,6)	(6,7)	(5,8)	(5,9)	(4,7)	(3,4)	(9,8)	(2,9)	
10	Subsets	2	1	0	0	0	0	0	0	0	0	0	0	0	1
5	$\bar{X} \subseteq N$	3	0	1	0	0	0	0	0	0	0	0	0	0	1
0		4	0	0	1	0	0	0	0	0	0	1	0	0	1
10		5	0	0	0	1	0	0	0	0	0	0	0	0	1
1		6	0	0	0	0	1	0	0	0	0	0	0	0	1
0		7	0	0	0	0	0	1	0	0	1	0	0	0	1
2		8	0	0	0	0	0	0	1	0	0	0	1	0	1
1		9	0	0	0	0	0	0	0	1	0	0	0	1	1
3		3,6,7	0	0	0	0	0	0	0	0	1	0	0	0	1
2		4,5,8	0	0	0	0	0	0	0	0	0	1	1	0	1
1		3,6,7,4,5,8	0	0	0	0	0	0	0	0	0	0	1	0	1
3		3,6,7,4,5,8,9	0	0	0	0	0	0	0	0	0	0	0	1	1
	$\bar{w}$		10	5	0	10	1	0	2	1	3	2	5	4	

Fig. 3.9.

References

[1] R.J. Duffin, "The extremal length of a network", *Journal of Mathematical Analysis and Applications* 5 (1962) 200.
[2] J. Edmonds, "Optimum branchings", in: *Mathematics of the decision sciences,* Eds. G.B. Dantzig and A.F. Veinott, Jr., Lectures in Applied Mathematics, Vol. 11 (American Mathematical Society, Providence, R.I., 1968) p. 346.
[3] J. Edmonds, "Submodular functions, matroids, and certain polyhedra", in: *Combinatorial structures and their applications,* Eds. R. Guy, H. Hanani, N. Sauer and J. Schonheim (Gordon and Breach, New York, 1970) p. 69.
[4] J. Edmonds, "Edge-disjoint branchings", to appear.
[5] L.R. Ford, Jr. and D.R. Fulkerson, *Flows in networks* (Princeton University Press, Princeto, N.J., 1962).
[6] D.R. Fulkerson, "Blocking polyhedra", in: *Graph theory and its applications,* Ed. B. Harris (Academic Press, New York, 1970) p. 93.
[7] D.R. Fulkerson, "Anti-blocking polyhedra", *Journal of Combinatorial Theory* 12 (1972) 50.
[8] D.R. Fulkerson, "Blocking and anti-blocking pairs of polyhedra", *Mathematical Programming* 1 (1971) 168.
[9] A. Lehman, "On the width–length inequality", mimeograph (1965).

Addendum

Since this paper was written, the author's attention has been directed to the following additional references, each of which treats the problem of constructing an optimum arborescence in a weighted directed graph.

Chu Yoeng-jin and Lin Tseng-hong, "On the shortest arborescence of a directed graph", *Scientia Sinica* 4 (1965) 1396.[M.R.33, #1245].
F.C. Bock, "An algorithm to construct a minimum directed spanning tree in a directed network", in: *Developments in operations research,* Ed. B. Avi–Itzhak (Gordon and Breach, New York, 1971) p. 29.
R.M. Karp, "A simple derivation of Edmonds' algorithm for optimum branchings", *Network* 1 (1971) 265.