# On DSN Antenna Scheduling

L. H. Harper, R. J. McEliece, and A. M. Odlyzko
Communications Systems Research Section

*We formulate and solve a problem related to the efficient assignment of DSN antennas to astronomical objects. The solution involves the well-developed theory of network flow.*

## I. Introduction

At any given time, the DSN may be required to observe simultaneously a large number of astronomical objects (spacecraft, planets, stars, comets, etc.). In general each object is not "visible" to each DSN site, and in addition it may be necessary to observe certain objects from more than one site, for example for interferometry experiments or for accurate tracking of accelerating spacecraft. It is the object of this article to give an efficient algorithm for assigning DSN antennas to astronomical objects under these circumstances. The method we shall give is derived from the theory of network flows (Ref. 1). Although we shall be concerned in this paper only with the static problem of DSN scheduling, in a future paper we will show how network flow theory can be used to solve dynamic scheduling problems, i.e., to take into account the fact that the demands on the network are time-varying.

Here is our general formulation of the static problem. We denote the antenna sites by $x_1, x_2, \cdots, x_n$, and suppose that the $i$-th site contains $h_i$ antennas. We suppose there are $m$ astronomical objects to be observed, and we denote by $A_j$ the subset of the antenna sites from which the $j$-th object is visible. We further suppose that the $j$-th object must be observed from $k_j$ different sites. A *solution to the assignment problem* is a collection of subsets of the antenna sites, say $B_1, B_2, \cdots, B_m$ ($B_j$ represents the sites from which the $j$-th object is to be observed) with the following property: $B_j$ contains $k_j$ elements, and no $x_i$ appears in more than $h_i$ of the $B_j$s. In the next section we shall give a necessary and sufficient condition for the assignment problem to have a solution, and give an efficient algorithm for finding a solution, if it exists.

## II. Solution to the Problem

Let $X = \{x_1, x_2, \cdots, x_n\}$ be a finite set, and let $A_1, A_2, \cdots, A_m$ be subsets of $X$; let nonnegative integers $h_1, h_2, \cdots, h_n; k_1, k_2, \cdots, k_m$ be given.

**Problem A:** Under what circumstances is it possible to find subsets $B_j \subseteq A_j$ $j = 1, 2, \cdots, m$, with $|B_j| = k_j$, and such that no $x_i$ appears in more than $h_i$ of the subsets $B_j$?

Let us observe first of all that there are many conditions which clearly must be satisfied in order for the problem to possess a solution. Thus suppose a solution $(B_j)$ exists, let $J$ be any subset of $\{1, 2, \cdots, m\}$ and denote by $B_J$ the multiset union $\bigcup_{j \epsilon J} B_j$. Furthermore define $h_i(J)$ by:

$$h_i(J) = \min(h_i, \sum_{j \epsilon J} c_{ij})$$

Clearly $x_i$ could occur no more than $h_i(J)$ times in the multiset $B_J$. Hence it follows that

$$\sum_{i=1}^{n} h_i(J) \geqq \sum_{j \epsilon J} k_j \qquad (1)$$

for all subsets $J$. Condition (1), then, is a *necessary* condition for the existence of a solution to our problem. It is a remarkable fact, however, that taken together these $2^m$ conditions are also *sufficient*:

**Theorem 1.** *If Condition (1) is satisfied for all subsets $J \subseteq \{1, 2, \cdots, m\}$, Problem A can be solved.*

**Proof.** This theorem is actually a special case of a known theorem on supply-demand networks. Rather than state the general theorem, however, we shall only describe the supply-demand network which applies to our particular problem.

The appropriate directed network has $n + m$ nodes; they are labeled $x_1, x_2, \cdots, x_n, A_1, A_2, \cdots, A_m$. There is a directed edge from $x_i$ to $A_j$ if and only if $c_{ij} = 1$, i.e., if $x_i \epsilon A_j$. Each of these edges is assigned a capacity of 1. The nodes $x_i$ are designated supply nodes; the supply at $x_i$ is $h_i$. The nodes $A_j$ are designated demand nodes; the demand at $A_j$ is $k_j$. A *flow* in this network is an integral-valued function $f_{ij}$ which satisfies $f_{ij} \leqq c_{ij}$.

One thinks of $f_{ij}$ as the amount of some quantity supplied to the demand node $A_j$ from the supply node $x_i$. The flow is said to be *feasible* if the demands are met and the supplies are not exceeded, i.e., if

$$\sum_{i=1}^{n} f_{ij} \geqq k_j \qquad j = 1, 2, \cdots, m$$

$$\sum_{j=1}^{m} f_{ij} \leqq h_i \qquad i = 1, 2, \cdots, n$$

Notice that such a feasible flow yields a solution to Problem A; merely take

$$B_j = \cup \{x_i : f_{ij} = 1\}$$

Conversely a solution to Problem A yields a feasible flow; define

$$f_{ij} = 1 \qquad \text{if } x_i \epsilon B_j$$
$$= 0 \qquad \text{if } x_i \epsilon / B_j$$

Now according to a theorem of Gale (see Ref. 1, Chapter 2: Theorem 1.1 and Corollary 1.2) a feasible flow exists if, corresponding to every subset of demand nodes, there is a flow that satisfies the aggregate demand of the subset without exceeding the supply limitations at each source. This result is clearly equivalent to Theorem 1.

Q.E.D.

While Theorem 1 is perhaps esthetically satisfying, it is clearly inefficient to check all $2^m$ conditions (1) in order to discover if Problem A has a solution. Even if one could check the conditions it is not clear how one could then actually construct a solution to the problem. A better way is to apply the "labeling algorithm" of Ford and Fulkerson (Ref. 1, Chapter 1) to the extended network corresponding to Problem A. This algorithm will quickly either produce a solution to Problem A, or a subset $J$ such that Condition (1) is violated.

The extended network is obtained from the network described in the proof of Theorem 1 by adjoining two additional nodes: "$s$" (the source) and "$t$" (the sink). There are edges from $s$ to each $x_i$; the capacity of the edge $(s, x_i)$ is $h_i$. There are also edges from each $A_j$ to $t$; the capacity of $(A_j, t)$ is $k_j$. In this extended network a flow is a function $f(a, b)$ on directed edges $(a, b)$ such that $f(a, b) \leqq c(a, b)$, the capacity of the edge $(a, b)$, and such that the net flow at each node other than $s$ and $t$ is 0; i.e.,

$$\sum_b f(a, b) = \sum_b f(b, a) \qquad \text{for all } a \neq s, t$$

The *value* of the flow is

$$v = \sum_b f(s, b) = \sum_b f(b, t)$$

The object of the labeling algorithm is to find a flow of maximum possible value. It turns out (see Ref. 1, Chapter 2) that a maximal flow in this extended network, when restricted to the original network, will be a feasible flow if a feasible flow exists, and in any case will be a flow which meets as many as possible of the demands without exceeding the supplies. If no feasible flow exists, the nodes $A_j$ which are *unlabeled* at the termination of the algorithm will be a subset violating Condition (1). Here is the algorithm: initially the flow is identically zero.

## Labeling Algorithm

(1) Give the sink $s$ the label $[\phi]$.

(2) (Scan $a$). Select any labeled, unscanned node $a$, with label $[c\pm]$. Give every unlabeled $b$, such that $f(a,b) < c(a,b)$, the label $[a+]$; give every unlabeled $b$, such that $f(b,a) > 0$, the label $[a-]$; $a$ is now "scanned."

(3) Repeat step 2 until either the sink $t$ is labeled [go to 4], or until no further labels can be assigned and $t$ is unlabeled [go to 6].

(4) Set $b = t$.

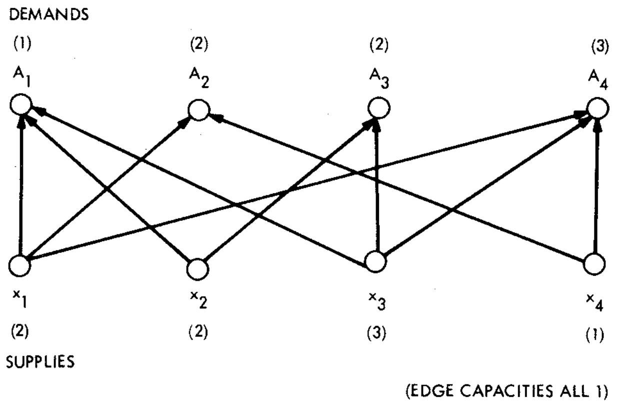
(5) If $b$ is labeled $[a+]$, increase $f(a,b)$ by 1; if $b$ is labeled $[a-]$, decrease $f(b,a)$ by 1. If $a = s$, erase all labels and go to 1; otherwise set $b = a$ and repeat step 5.

(6) Either all demands $h_i$ are satisfied and Problem A is solved, or the set of unlabeled $A_i$s have an aggregate demand in excess of the total supply, and the problem is unfeasible.

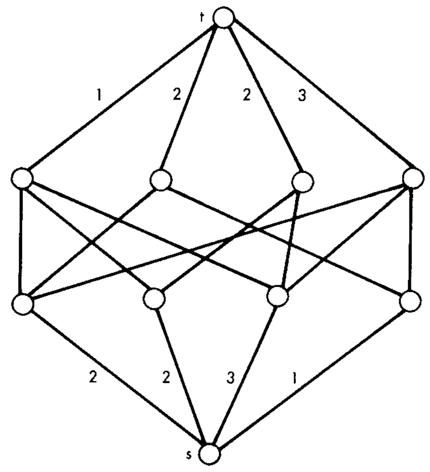
The labeling algorithm is quite efficient: notice that since this value of the flow is increased by 1 each time step 5 is executed, and that during the labeling process each edge needs to be examined at most twice, the total work involved is at most

$$2 \cdot \sum_{j=1}^{m} k_j \cdot \sum_{i,j} c_{ij} \leq 2m^2 n\, k_{\max} \qquad \text{if } k_{\max} = \max(k_j)$$

We conclude with an example which illustrates these techniques. Let $n = m = 4$, $A_1 = \{x_1, x_2, x_3\}$, $A_2 = \{x_1, x_4\}$, $A_3 = \{x_2, x_3\}$, $A_4 = \{x_1, x_3, x_4\}$; $h_1 = 2$, $h_2 = 2$, $h_3 = 3$, $h_4 = 1$; $k_1 = 1$, $k_2 = 2$, $k_3 = 2$, $k_4 = 3$. The corresponding supply-demand network is



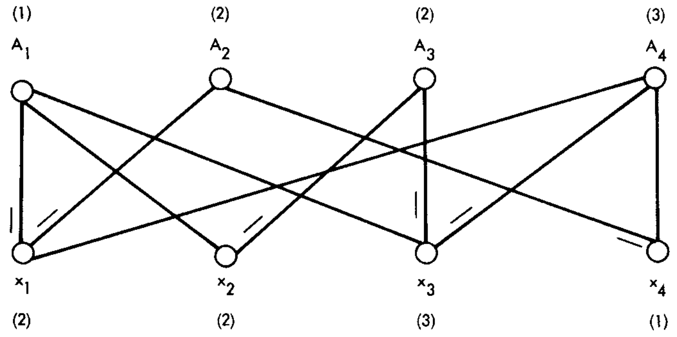
and the extended network is



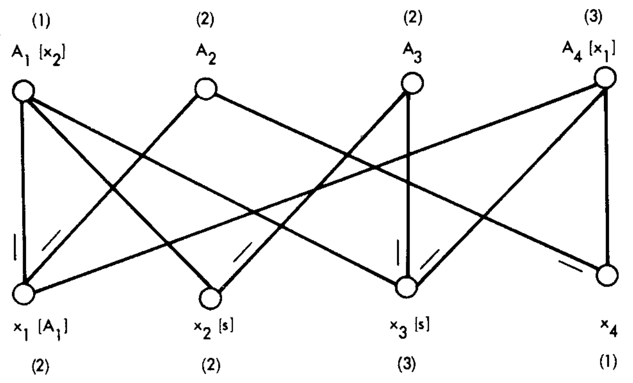
We now apply the labeling algorithm to the extended network. For the purposes of this example, we will always scan a labeled $A_j$ if the sink $t$ can be labeled from $A_j$, i.e., if $f(A_j, t) < k_j$. Otherwise our scanning priority will be $(x_1, x_2, x_3, x_4, A_1, A_2, A_3, A_4)$. The problem is solved in 9 passes through the algorithm, which we list below:

| Pass number | Order in which nodes are scanned |
|---|---|
| 1 | $s, x_1, A_1, t$ |
| 2 | $s, x_1, A_2, t$ |
| 3 | $s, x_2, A_3, t$ |
| 4 | $s, x_2, x_3, A_3, t$ |
| 5 | $s, x_2, x_3, A_4, t$ |
| 6 | $s, x_2, x_3, x_4, A_2, t$ |
| 7 | $s, x_2, x_3, A_1, x_1, A_4, t$ |
| 8 | $s, x_2, A_1, t$ |
| 9 | $s, x_3, A_1, x_2$ |

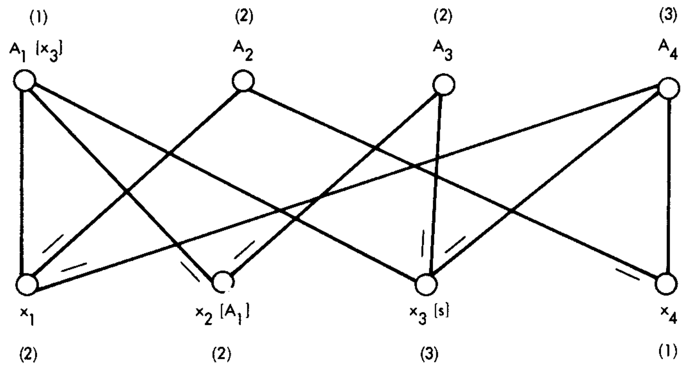
Notice that pass 7 involves backtracking, viz., $f(x_1, A_1)$ is decreased. Immediately prior to the execution of pass 7, the network looks like this (omitting the nodes $s$ and $t$):



(An edge $(x_i, A_j)$ with $f(x_i, A_j) = 1$ is indicated by a short dash parallel to that edge.) After the labels have been added, and the sink $t$ has been reached via $A_4$, the network looks like this:



(We have omitted the $\pm$ signs on the labels, since labels on the $x_i$ are always either $[s-]$ or $[A_j+]$ and those on the $A_j$ are always $[x_i-]$.) The execution of step 5 now increases $f(A_4, y)$ by 1, increases $f(x_1, A_4)$ by 1, decreases $f(x_1, A_1)$ by 1, increases $f(x_2, A_1)$ by 1, and increases $f(s, x_2)$ by 1. After this the algorithm runs smoothly, and ends in the following configuration:



No further labels can be assigned, since all labeled nodes are also scanned. Thus the illustrated flow is maximal, but the demand at $A_4$ is unsatisfied. The unlabeled $A_j$s $\{A_2, A_3, A_4\}$ have an aggregate demand of 7, whereas the total supply into $\{A_2, A_3, A_4\}$ is $2 + 1 + 2 + 1 = 6$, and so the problem is palpably unfeasible. Notice that if $k_j = 2$ instead of 3 the algorithm would have terminated with all demands being met.

## Reference

1. Ford, L. R., and Fulkerson, D. R., *Flows in Networks,* Princeton University Press, Princeton, New Jersey, 1962.