# TRAFFIC ASSIGNMENT IN COMMUMICATIOM SATELLITES

by

E. Balas •* P.P. Landweer

December, 19?3

TRAFFIC ASSIGNMENT IN

COMMUNICATION SATELLITES

by

Egon Balas

and

Philip R. Landweer

April 1983
Revised June 1983

# Abstract

A high capacity communication satellite interconnects scores of ground stations simultaneously. Under the Satellite-Switched/ Time Division Multiple Access (SS/TDMA.) system, each channel of the satellite is allocated to a pair of ground stations for a certain time period, after which the whole set of allocations (called a switch) is changed simultaneously. The problem we address is to minimize the time length of the entire sequence of switches, subject to a limit on the number of switches. We formulate this as a 3-index bottleneck-sum assignment problem, and solve it by a heuristic that obtains consistently better results than earlier methods based on different formulations.

## 1. Introduction: Background and History

Every day a rapidly increasing volume of long distance TV, radio and telephone communications is transmitted digitally via satellites. A high capacity communication satellite interconnects simultaneously scores of transmitting and receiving stations. One of the advanced techniques for operating such a satellite communication network is the SS/TDMA (satellite-switched time-division multiple access) system, based on the use of highly directive spotbeam antennas [12], [1]. Under this scheme, each transponder on board of the satellite is allocated to a pair of zones (groups of ground stations) for a certain amount of time. Such a set of allocations involving all the transponders is called a _switch mode_. The allocations that make up a switch mode are changed simultaneously by an on-board switching facility. The whole sequence of switch modes that make up the schedule of allocations for a given time period is called a _frame_.

A major problem that arises in connection with SS/TDMA systems is the efficient scheduling of time slot allocations for a frame. To be more specific, the demand for a frame can be expressed as an n×n matrix $D = (d_{ij})$, the _traffic matrix_, where $d_{ij}$ represents the length of the data burst from zone i to zone j, i.e., the amount of time for which a transponder needs to be assigned to the pair of zones (i,j). Clearly, $d_{ij} \geq 0$ for all i,j. The scheduling task consists of allocating time slots on the satellite's transponders to pairs of zones. More specifically, it consists of decomposing the traffic matrix D into n×n switch matrices (or mode matrices) $D^k = (d_{ij}^k)$, k = 1,...,q, subject to certain feasibility conditions and satisfying some efficiency criterion. The feasibility conditions are as follows.

For k=1,...,q, every row or column of $D^k$ contains at most one nonzero element. (1)

in other words, in every switch mode, each transponder is assigned to at most one pair of zones.

$$\sum_{k=1} Z \, D^* \gg D, \tag{2}$$

i.e., the q switch modes together meet total demand.

It is easy to see that for $q < n$ the problem is infeasible whenever D has at least one row or column whose entries are all positive. On the other hand, for $q * n$ the problem amounts to 1-factorizing the complete bipartite graph $K_{n,n}$ (i.e., finding in $K_{n,n}$ n pairwise edge-disjoint perfeet matchings), a problem whose feasibility is an immediate consequence of the König-Hall theorem on perfect matchings [7]. Thus for $q \geq n$ the problem is always feasible.

As to the efficiency criterion, a first objective is to satisfy total demand in a minimum amount of time. Since the time needed to transmit one switch mode is given by the largest entry of the switch matrix $D^k$, total transmission time for a given solution $(D^1,...,D')$ is

$$T - \sum_{k=1}^{q} \max_{i,j} d^k_{ij}. \tag{3}$$

Thus one approach is to address

<u>Problem 1</u> [9]. Find a schedule $(D^1,...,D^q)$ that satisfies (1), (2) and minimizes (3).

The <u>value</u> of an optimal solution to this problem can be found out by inspection. First, it is not hard to see that any row sum or column sum of D is a lower bound on the value of T; hence the same is true of the maximum of these row and column sums, i.e., $T \geq T^*$, where

$$T^* \ll \max \left\{ \max_i \sum_{j=1}^{n} d_{ij}; \ \max_j \sum_{i=1}^{n} d_{ij} \right\}. \tag{4}$$

Second, it has been shown [9, 3] that (i) ruin T * T*, (ii) there exists an optimal solution (D $_{f1i}$.,D0 with $q < Q" - 2n \wedge 2$; and (iii) this is the smallest bound on the value of q for which the existence of a schedule with T * T* can be guaranteed.

While (iii) holds for an arbitrary traffic matrix D, in the not in-frequent case when D con'tains some entries equal to $0_9$ the bound on q can be strengthened to $q < n^2 - 2n + 2 - z$, where z is the number of elements of D equal to 0.

Polynomial time procedures for finding an optimal solution to Problem 1 are given in [9, 8].

A second criterion one may want to consider has to do with the size of q, the number of switch modes into which the traffic matrix is decom-posed. Theoretical considerations as well as practical experience indi-cate that for an optimal solution to Problem 1, i.e. for a schedule $(D^1,...,D^Q)$ with T • T*, the value of q is typically equal to or close to its upper bound, $n^2 - 2n + 2 - z$. This is inconvenient for the following reason. A schedule involving q switches requires as many switch reconfigurations. Further, if q > a, some data bursts must be split, i.e., only partially transmitted during a switch mode, with the rest transmitted in another switch mode. Switch reconfigurations and burst splittings entail che addition of preambles and guard margins, not to mention other inconveniences. Therefore it is essential to keep q as small as possible, and the second relevant problem in this context can be formulated as

Problem 2 [2]. Find a schedule $(D^1,...,D^q)$ chat satisfies (1), (2) and q = n, and minimizes T.

The practical importance of Problem 2 comes from the fact that it avoids burst splittings. However, its relevance depends on how close its optimum is to T*. la this context, the (transmission) <u>efficiency</u> of a schedule with total transmission time T is defined as the ratio T*/T. Currently no bound is known on the efficiency of an optimal solution to Problem 2. Furthermore, as discussed below, Problem 2 is NP-complete and for realistic matrix sizes too large to be solved to optimality. Thus one has to look for approximate solution methods. Naturally, if no bound is known on the efficiency of an optimal solution, even less is known theoretically about the efficiency of approximate solutions. However, some empirical evidence exists in this respect, Camerini, Maffioli and Tartara [2] (in the following CMT) have developed and tested a heuristic for Problem 2, that consists of n successive applications of the following iterative step (which starts with k = 1):

Let  N = {1,...,a}, and

$$X = \{x \in K^n \mid \sum_{j=1}^{n} x_{ij} = 1, i\in N; \sum_{i=1}^{n} x_{ij} = 1, j\in N; x_{ij} \in \{0.1\}, i,j\in N\} \quad (5)$$

<u>Iterative Step</u>.  Solve the nxn assignment problem

$$(AP) \quad \max_{x\in X} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$$

and, if x* is the optimal solution found, let $d_{ij}^k = d_{ij}$ if $x^*_{ij} = 1$, $d^*_{ij} := 0$ if $x^*_{ij} = 0$, i,j\in N.  Set $d_{ij} := -\infty$ if $x^*_{ij} = 1$, $d_{ij} := d_{ij}$ if $x^*_{ij} = 0$, for i,j\in N. Then if k < n, set k:=k + 1 and repeat the step.  Otherwise $(D^1,...,D^n)$ is the desired schedule.

The assignment problem is solved at each iteration by the Hungarian method [10] (for a more recent treatment, see [3] or [11]), whose time complexity is $O(n^3)$. Thus the time complexity of the whole procedure is $O(n^4)$. As to its transmission efficiency, on a set of 100 test problems with $n \ll 20$ and the entries of D integers drawn randomly from a uniform distribution over the interval [0, 100], it was found to be on the average about 89-90%, with a minimum of 83% and a maximum of 96%.

In a way, Problems 1 and 2 above represent two extremes, in which one of the two criteria of interest is driven to its minimum value without regard to what this entails in terms of the other. In order to examine some intermediate situations, CMT [2] have also addressed the following class.

Problem 3. For $h = 1,2,\ldots$, find a schedule $(D^1,\ldots,D^a)$ that satisfies (1), (2) and $q \ll hn$, and minimizes T.

Finally, there is a substantial difference between burst splittings that do and do not separate bursts into nonadjacent time slots, the former (called proper burst splittings) being more costly than the latter. In order to reflect this differentiation, CMT [2] have formulated

Problem 4. For $h = 1,2,\ldots$, find a schedule that satisfies (1), (2) and $q = hn$ without proper burst splittings, and minimizes T.

CMT have modified their heuristic to obtain procedures of the same complexity ($O(n4)$) for finding approximate solutions to Problems 3 and 4. 3.un on the same test problems, these procedures tended to produce solutions of improved efficiency as h was increased.

## 2. A New Heuristic for Problem 2

The CMT heuristic is based on the idea that for any k, maximizing the sun of elements of $D^k$ will force inco $D^k$ as many of the larger eleser.cs

of D as possible, thus reducing the value of the largest element left in $D^{k-1}$ . In order to gain some insight into what might be a better heuristic, it is useful to look at some alternative formulations of Problem 3. The first formulation is a 3-index bottleneck-sum assignment problem with $3n^2$ constraints and $n^3$ variables, in which $d_{..}$ is represented as $d_{..}x_{..k}$, with $x_{ijk} = 0$ or 1:

$$\min_{x} \sum_{k=1}^{n} \max_{i,j} d_{..} x_{ijk}$$

$$\sum_{j=1}^{n} x_{ijk} = 1 \qquad i,k = 1,\ldots,n$$

$$\sum_{i=1}^{n} x_{ijk} = 1 \qquad j,k = 1,\ldots,n \qquad \text{(P2.1)}$$

$$\sum_{k=1}^{n} x_{ijk} \le 1 \qquad i,j = 1,\ldots,n$$

$$x_{ijk} \in \{0,1\} \qquad i,j,k = 1,\ldots,n$$

The second formulation is a set partitioning problem with $n^2$ equations and $n!$ variables, one variable associated with every $n \times n$ permutation matrix:

$$\min cy$$

$$Ay = e \qquad \text{(P2.2)}$$

$$y_j \in \{0,1\}, \quad j = 1,\ldots,n!$$

Here $e = (1,\ldots,1)^T$ has $n^2$ components, and every column $a_j$ of A represents an $n \times n$ permutation matrix in vector form, i.e., $a_{ij} \in \{0,1\}$, $i = 1,\ldots,n^2$, with

$$\sum_{r=1}^{n} a_{(r-1)n+s,j} = 1 , \qquad s = 1,\ldots,n$$

$$\sum_{s=1}^{n} a_{(r-1)n+s,j} = 1 , \qquad r = 1,\ldots,n;$$

and all the columns of A are distinct. Further,

$$c_j = \max_{1 \le r,s \le n} a_{(r-1)n+s,j} d_{rs} .$$

Both (P2.1) and (P2.2) are NP-complete problems which one cannot hope to be able to solve to optimality except for small values of n. But they both offer some insights into the kind of heuristics that might be expected to work. In particular (P2.1) suggests that instead of solving a sequence of maximizing assignment problems $(AP)_k$, one might do better by solving a sequence of bottleneck, or min-max assignment problems. This leads to the following:

<u>Min-max Procedure.</u>

Let X be given by (5). Set k = 1 and go to the

<u>Iterative Step</u>. Solve the nxn bottleneck assignment problem

$$(BAP)_k \quad \min_{x \in X} \ \max_{i,j} \ d_{ij} x_{ij}$$

and if x* is the optimal solution found, define $d_{ij}^{k} := d_{ij}$ if $x_{ij}^{*} = 1$, $d_{ij} := 0$ if $x_{ij}^{*} = 0$, $i,j \in N$. Set $d_{ij} := \infty$ if $x_{ij}^{*} = 1$, $d_{ij} := d_{ij}$ if $x_{ij}^{*} = 0$, for all $i,j \in N$. Then if $k < n$, set $k := k + 1$ and repeat the step. Otherwise $(D^1,\ldots,D^n)$ is the desired schedule.

On the other hand, the set partitioning formulation (P2.2) suggests the use of the greedy algorithm for set covering problems [4], modified to handle equations instead of inequalities. Since the column sum $\sum_i a_{ij}$ is the same for every column of A, and a cover can only contain pairwise orthogonal columns, the greedy choice rule is equivalent in this case to minimizing $c_j$. Thus the greedy algorithm adapted to (P2.2) amounts to sequentially choosing columns $a_j$ of A in order of increasing $c_j$, while deleting after each choice all columns $a_k$ such that $a_k a_j \geq 1$, until a set of n columns has been chosen. But in view of the definition of the $a_j$ and $c_j$, this is precisely the Min-max Procedure stated above. Thus both problem formulations suggest the same heuristic approach.

The Iterative Step of the Min-max Procedure can be solved by the "threshold method[11]" of O. Gross [6] (see also [5] and [11]). In the nXn array associated with the (bottleneck) assignment problem, call a row or column a <u>line</u>. Call a set S of cells (elements of the array) <u>independent</u>, if no two cells of S are in the same line. An independent set of cardinality n is an <u>assignment</u>. The "threshold method" starts with some initial assignment $S_0$, defines the threshold

$$\delta := \max\{d_{ij} \mid (i,j) \in S_0\}$$

and declares <u>admissible</u> those cells $(i,j)$ such that $d_{ij} < \delta$. It then uses a labeling procedure to find a maximal independent set S of admissible cells. If $|S| < n$, then $S_0$ defines an optimal solution $x^0$ to (3A?), with $x^0_{ij} = 1$ if $(i,j) \in S_0$, $x^0_{ij} = 0$ otherwise. If $|S| = n$, the threshold can be lowered: one sets $S_0 := S$, redefines $\delta$, and repeats the procedure.

We have slightly aodified this basic procedure in order to use the fact that we are solving a sequence of interrelated bottleneck assignment problems rather than just one. The main change is that, rather than lowering the threshold each time to the value of the next largest element of D, we calculate an upper and a lower bound on the objective function value and :hoose the threshold via bisection of the interval defined by these bounds. This has speeded up the procedure considerably.

To generate a starting assignment, we use a "stingy" heuristic that chooses admissible cells in order of increasing $d_{ij}$. If we have to stop short of choosing n cells, we complete the assignment via the labeling procedure. The details follow.

A line of the assignment array will be called <u>free</u> if it has no cell in the current assignment. At the start D is the traffic matrix, sub-sequently D is updated at the end of each iteration. Set k := 0 and go to the

### Iterative Step

1.  (Construct Initial Assignment)

If k = 0, set 9:*0 and go to 1.2. Otherwise let D" be the last switch matrix produced; set 3:*max d.., declare admissible all cells (i,j) such that $d_{ij}$ < 8, and go to 1.1.

1.1. Scan the rows and put into S the first admissible cell in a free column encountered in each row. If |S| * n go to 2, else continue.

1.2. Find $d_i$ . .« ad.rx[d..ji and j are free}. If d. . < a, $_{?uc}$ (i ,j ) into S and go to 1.2; else continue.

1.3. If |s| » n, go to 2; else declare all cells admissible and complete the assignment by the labeling procedure: set 5(?) := 0 and go to 3.

2. (Sec New Threshold)

Set $S^* := S$, store the candidate assignment $S^*$, and define

$$\overline{\vartheta} = \max\{d_{ij} \mid (i,j) \in S\},$$

$$\underline{\vartheta} :* \max\{9, \min_{i,j} d_{ij}\},$$

and

$$\delta(\vartheta) := \begin{cases} 1 & \text{If } \overline{\vartheta} - \underline{\vartheta} \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

**Then set**

$$\vartheta := \lfloor \tfrac{1}{2}(\overline{\vartheta} + \underline{\vartheta}) \rfloor,$$

$$S := S \setminus \{(i,j) \in S \mid d_{ij} \geq \vartheta\},$$

declare admissible all cells $(i,j)$ such that $d_{ij} < \vartheta$, and go to 3.

3. (Find a Maximal Independent Set of Admissible Cells)

Start with all lines unlabeled, unscanned.

3.1-3.3. (Labeling)

3.1. Label all free rows with -1 and go to 3.2.

3.2. Choose a labeled unscanned row i and scan it for admissible cells. For each such cell $(i,j)$, if column j is unlabeled, label it with i. Then if column j is free, let $j_0 := j$ and go to 3.4, else continue.

Repeat 3.2 until all labeled rows have been scanned. If no new column has been labeled, go to 3.5; else go to 3.3.

3.3. Choose a labeled unscanned column j and scan it for some $(i,j) \in S$. If such $(i,j)$ is found, label row i with j.

Repeat 3.3 until all labeled columns have been scanned. If no *new* row has been labeled, go to 3.5; else go to 3.2.

3.4.  (Augmentation)  If the label of column $j_0$ is i, put $(i,j_0)$ into S. If the label of row i is $j * -1$, make all lines unlabeled, unscanned, and go to 3.1.  If the label of row i is $j ^ -1$, put $(i,j)$ into S, set $j_0 :* j$ and go to 3.4.

3.5.  S is a maximal independent set of admissible cells.

If $js| * n$ and $5(3) * 0$, the threshold can be lowered: go to 2.

If $|s| \ll n$ and $5(3) * 1$, S is a min-max assignment:  set $S* :\ll S$ and go to 4.

If $|s| < n$, no assignment exists that satisfies the current threshold. Therefore $S*$, the candidate stored in step 2, is a min-max assignment:  go to 4.

4.  (Construct New Switch Matrix)

Define $D^{k+1} * (dj_j^1)$, where

$$d_{ij}^{k+1} := \begin{cases} d_{ij} & (i,j)eS* \\ 0 & (i,j)\&S*, \end{cases}$$

and set $d_{\pm \cdot j} :\gg \gg$ if $(i_fj)eS*$, $d_{t \cdot j} :*<^*_{ij}$ otherwise.  If $k + 1 \bullet a$, stop: $(D^1,...,D^n)$ is the desired schedule.  Otherwise set $k :^s k + 1$ and go to 1.

A few comments are in order.  We will call an application of the Iterative Step a cycle.  At the beginning of each cycle, the construction of an initial assignment starts by putting into S only cells that are admissible with respect to the threshold value of the previous iteration (Step 1.1).  Step 1.2 continues the construction by including additional cells according to the "stingy[11] rule of choosing them in order of increasing $d_{ij}$.  When no complete assignment can be obtained in this way, the labeling procedure of Step 3 is used to finish the job, with no restriction on admissibility.

Step 2 is entered with a complete assignment S. Clearly, $\max_{i,j} \{d_{ij} \mid (i,j) \in S\}$ constitutes an upper bound on the value of the bottleneck objective function for the switch matrix under construction, and $\max\{\bar{s}, \min_{i,j} d_{ij}\}$, where $\bar{s}$ is the threshold value of the previous iteration, constitutes a lower bound. To see this latter point, note that if a switch matrix with an objective function value lower than $\bar{s}$ existed, it would have been found at the previous iteration. A new threshold $\bar{s}$ is then defined by bisecting the interval between the above upper and lower bounds. The role of the parameter $\delta(\bar{s})$ is to carry to Step 4 the instruction of either continuing the bisection procedure, or terminating it.

Step 3 is the well-known labeling procedure of Ford and Fulkerson for finding a maximal independent set of admissible cells. Steps 2 and 3 are iterated as long as feasible assignments can be found for successively lower thresholds. This process stops when lowering the threshold is no longer possible. At that point the current assignment S* is used to construct the next switch matrix $D^{k+1}$, the traffic matrix D is updated, and a new cycle is started. After a total of n cycles, the procedure ends with a feasible schedule $(D^1, \ldots, D^q)$, $q = n$.

Next we address the computational complexity of the algorithm. During every cycle, Step 1 is used exactly once, and its time complexity is $O(n^3)$. Step 2 is $O(n^2)$. The labeling procedure of Step 3 is $O(n^2)$, each augmentation is $O(n)$, and the number of augmentations (and hence labelings) during one application of Step 3 is $O(n)$. Hence one application of Step 3 has a total time complexity of $O(n^2)$. Finally, Steps 2 and 3 can be iterated during a cycle at most $\log_2 \Delta$ times, where $\Delta = \max_{i,j} d_{ij} - \min_{i,j} d_{ij}$. Thus the time complexity of a cycle is $O(n^3 \log_2 \Delta)$, and since there are n cycles, the time complexity of the whole Min-max procedure is $O(n^4 \log_2 \Delta)$.

Lawler [11] gives a version of the bottleneck assignment algorithm whose time complexity is $0(n^3)$; so when that algorithm is substituted for Steps 2-3 above, the resulting procedure is of $0(n^4)$. We have implemented and extensively tested both procedures. In spite of its better worst case bound, the version using Lawler's algorithm was on the average about 30% slower than the Min-max Procedure using the Ford-Fulkerson method combined with bisection.

### 3. Computational Results

Both the CMT method [2] and the Min-max Procedure were implemented and extensively tested on a VAX computer, on randomly generated test problems with the elements of D drawn from a uniform distribution of the integers over [1,100]. A total of 12 classes of problems were solved, with 1000 problems in each of the first 6 classes (for n « 5, 10, 15, 20, 30 and 40), and 20 problems in each of the next 6 classes (for n * 50, 60, 70, 80, 90 and 100). Table 1 summarizes the results.

Table 1.  Computational Results on a VAX 811

(1000 problems in each of the first 6 classes,
20 problems in each of the last 6 classes)

| a | Average efficiency (%) | |
|---|---|---|
| | CUT | Min-max |
| 5 | 91.46 | 93.58 |
| 10 | 88.57 | 92.73 |
| 15 | 37.62 | 92.71 |
| 20 | 90.78 | 95.12 |
| 30 | 88.63 | 94.81 |
| 40 | 89.57 | 96.07 |
| 50 | 91.22 | 97.39 |
| 60 | 90.37 | 97.69 |
| 70 | 90.27 | 97.53 |
| 80 | 90.39 | 97.44 |
| 90 | 91.21 | 98.23 |
| 100 | 91.43 | 98.25 |

The performance of the Min-max Procedure in terms of transmission efficiency is uniformly better than that of the earlier CMT heuristic, and increasingly so for large problems. While the efficiency of the CMT method fluctuates around 89-91% for small as well as for large problems, the efficiency of the Min-max Procedure is around 93% in the range $5 \leq n \leq 15$, around 95% for $20 \leq n \leq 40$, around 97.5% for $50 \leq n \leq 80$, and above 98% for $90 \leq n \leq 100$.

As to the computational effort, we have found that while the time used by the CMT method is $O(n^4)$, i.e., close to its upper bound, the time required by the Min-max Procedure only slightly exceeds $O(n^3)$, in spite of the somewhat weaker worst case bound.

With transmission efficiencies so close to 1, it seems that restricting the number of switch matrices to n is not such a debilitating constraint as it had been thought to be. In the light of this computational experience, Problem 2 of Section 1, first formulated by CMT [2], appears to be the right problem to address, and the Min-max Procedure recommends itself as the method of choice.

The Min-max Procedure, just like the CMT method, can be modified to solve Problems 3 and 4 stated in Section 1. We have tested such modified versions of our procedure and obtained for Problem 3 average transmission efficiencies of 97.86% for $q = 2n$, 98.58% for $q = 3n$ and 99.12% for $q = 5n$, on 100 problems with $n = 20$. These are better than the 95.12% efficiency obtained for Problem 2 (i.e., for $q = n$), but we find it doubtful whether this further improvement can compensate for the disadvantage of having burst splittings.

la the case when proper burst splittings are forbidden (Problem 4), we managed to obtain only a very slight improvement over the efficiency registered for Problem 2 (95.14% for q -2n$_f$ 95.39% for q » 3n, and 95.83% for q » 5n), which seems to rule out Problem 4 as a promising formulation.

Finally, returning to Problem 2, we have also tested a procedure that combines the Min-max heuristic with a limited amount of implicit enumeration in order to find better solutions at increased computational cost. To be more precise, this procedure generates the n alternative solutions obtainable by banning from D$^1$ (the first switch matrix generated by the Min-max heuristic) one of its n positive entries, and then applying the procedure as usual. At the cost of an n-fold increase in computing time, the average improvement in efficiency was 0.21%, 0.23%, 0.22% and 0.18% for the 50 problems with n « 5, 10, 15 and 20 respectively. These meager improvements strongly suggest that the solutions to Problem 2 obtained by the Min-max Procedure are very close to the optimum.

## Acknowledgments

## References

[1]  H. Arnold, "An Efficient Digital Satellite Technique for Serving Users of Differing Capacities." <u>International Conference on Communication</u>, Chicago, 1977, p. 6.1-116-119.

[2]  P. Camerini, F. Maffioli and P. Tartara, "Some Scheduling Algorithms for SS/TDMA Systems/' <u>Fifth International Conference on Telecommunications</u>« Genoa, 1981, p. 405-409.

[3]  N. Christofides, <u>Graph Theory: An Algorithmic Approach</u>. J. Wiley, 1975.

[4]  V. Chvatal, "A Greedy Heuristic for the Set Covering Problem." <u>Mathematics of Operations Research</u>; 4, 1979, p. 233-235.

[5]  L. R. Ford and D. R. Fulkerson, <u>Flows in Networks</u>. Princeton University Press, 1962.

[6]  O. Gross, "The Bottleneck Assignment Problem: An Algorithm." <u>Proceedings of the RAND Symposium on Mathematical Programming</u>, RAND Publication R-351, 1960, p. 37-88.

[7]  P. Hall, "On the Representation of Subsets." <u>Journal of the London Mathematical Society</u>, 10, 1935, p. 26-30.

[8]  T. Inukai, "An Efficient SS/TDMA Time Slot Assignment Algorithm." <u>IEEE Transactions on Communications</u>, COM 27, 1979, p. 1449-1455.

[9]  Y. Ito, Y. Urano, T. Muratani, M. Yamaguchi, "Analysis of a Switch Matrix for an SS/TDMA System." <u>Proceedings of the IZZE</u>, 65, 1977, p. 411-419.

[10]  H. W. Kuhn, "The Hungarian Method for the Assignment Problem." <u>Naval Research Logistics Quarterly</u>, 2> 1955, p. 83-97.

[11]  E. L. Lawler, <u>Combinatorial Optimization: Networks and Matroids</u>. Holt, Rinehart and Winston, 1976.

[12]  W. Schmidt, "An On-Board Switched Multiple Access System for Millimeter Wave Satellites." <u>International Conference on Dizital Satellite Communication</u>, London, 1969, p. 399-406.