Time Slot Assignment in SS/TDMA Systems with Intersatellite Links

ALAN A. BERTOSSI, GIANCARLO BONGIOVANNI, AND MAURIZIO A. BONUCCELLI

Abstract—**In this paper we study the time slot assignment problem in clusters of SS/TDMA satellite systems interconnected through intersatellite links. We show that the problem of finding an assignment which minimizes the total transmission time is NP-complete, i.e., computationally intractable, even for quite restricted intersatellite link patterns and simplified system models. Successively, we focus our attention on clusters of two satellites, proposing a branch-and-bound optimal algorithm and two fast heuristic algorithms. We investigate the performance of the proposed heuristic algorithms both by a theoretical worst case bound and by simulation trials showing that the produced solutions are close to the optimal on the average.**

I. Introduction

DEMANDS of satellite communications services are rapidly growing, and the natural resources they use, the RF spectrum and the geosynchronous orbit, are becoming highly crowded.

A more efficient use of the electromagnetic spectrum can be achieved by using multibeam antennae and satellite-switched time-division multiple-access (SS/TDMA) techniques [1]–[22]. In such a case, the satellite has a number of spot beam antennae covering several geographical *zones* and a solid-state RF switch on board to allow interconnectivity between the various uplink and downlink beams. The TDMA transmission is made up of frames, divided in subframe intervals, or *time slots*. Each time slot represents a particular switching configuration, which allows to transmit a certain amount of traffic between the connected uplink and downlink beams.

In many practical situations, ground stations exchanging traffic are not always visible by the same satellite. In such a case, the current practice is either to use ground communication lines or to reroute the traffic through an intermediate ground station in the line of sight of two satellites, one visible by the transmitter and the other visible by the receiver. In both cases, extra earth resources are used, thus reducing the total efficiency of the system.

A solution to this problem which does not require the introduction of additional satellites is given by interconnecting the two satellites by an *intersatellite link* (ISL, for short) [22]–[27]. ISL's have the additional advantage of allowing several small and less expensive satellites to join their coverage and capability so to have the communication power of a much larger and more expensive satellite. In the next decade, pairs of communication satellites interconnected via ISL's will probably become operational [22]–[25], [27]. Large clusters of satellites with various ISL patterns to form a global communication network are expected in a more distant future [25]–[26].

The combination of the above techniques, i.e., SS/TDMA satellites interconnected via ISL's, is very promising. The performance of these systems depends on several factors. A very important one is a proper assignment of traffic to time slots so that transmission *conflicts* are avoided. Specifically, no more than one transmitter can send traffic to the same destination simultaneously and no transmitter can send traffic to more than one destination simultaneously; moreover, for every two satellites interconnected by an ISL, no more than one transmitter in the line of sight of each satellite can send traffic through the ISL to any destination in the line of sight of the other satellite at the same time. The objective is then of scheduling all the traffic in time slots with the maximum transponder utilization, which in turn can be achieved by minimizing the overall duration of the schedule.

Inukai [22] investigated the ISL time slot assignment problem for clusters of two SS/TDMA satellites with onboard *buffers*. He showed how to reduce the scheduling problem for such a system configuration to the widely studied single-satellite scheduling problem [6]–[21]. In fact, the traffic transmitted over the ISL can be stored by the receiver satellite into the ISL buffer, and sent to the destination zone when no conflict may arise.

In the present paper, we study the time slot assignment problem for unbuffered satellites. In this case, all the traffic received by a satellite must immediately be sent to ground zones and over the ISL, according to the configuration of the onboard RF switch. Since the satellites connected by ISL's are physically separated, the traffic transmitted over the ISL reaches the destination satellite after a propagation delay δ. In this way, if zone i sends a message to zone j and the switch on the transmitter satellite is set to connect zone i with ISL at time t, the switch on the receiver satellite must be set to connect ISL with zone j at time $t + \delta$. The effects of δ on the scheduling problem depend on its magnitude in comparison to the duration of a time slot.

Throughout this paper, we study the simpler scheduling problem given by assuming δ negligibly small with respect to time slot duration. In particular, we show that even this simplified scheduling problem is computationally intractable and *a fortiori* the general problem with nonnegligible δ is hard to be solved. A formal definition of the problem we are dealing with is given in Section II. In Section III we study clusters of an arbitrary number of satellites and show that the problem is NP-complete even for quite restricted intersatellite link patterns. In Section IV, we consider the case of clusters comprising exactly two satellites. We propose two fast suboptimal heuristic algorithms, both generating very close to optimal schedules on the average, along with a time consuming optimal branch-and-bound algorithm.

Finally, we assume in this paper that each satellite in the cluster covers the same number N of zones and is provided with at least $N + k$ transponders if it is connected to k satellites via ISL's. Besides, we assume that all uplink, downlink, and ISL beams have equal bandwidth, and that intersatellite links allow transmission of traffic from one

Paper approved by the Editor for Satellite Communications and Coding of the IEEE Communications Society. Manuscript received December 13, 1985; revised April 17, 1986. This work was supported by a grant from the Ministry of Public Instruction, Italy. This paper was presented at GLOBECOM '85, New Orleans, LA, December 1985.

A. A. Bertossi and M. A. Bonuccelli are with the Department of Computer Science, University of Pisa, Pisa 56100, Italy.

G. Bongiovanni is with the Department of Mathematics, University of Rome, Rome 00100, Italy.

IEEE Log Number 8714586.

0090-6778/87/0600-0602$01.00

satellite to the other one and vice-versa. We also assume that no zone is covered by more than one satellite.

II. Mathematical Formulation

An *ISL traffic matrix* D for a cluster of s satellites is an $sN \times sN$ matrix with nonnegative integer entries. Entry d_{ij} in D represents the amount of traffic that zone i must transmit to zone j. The number of zones is sN. The zones are partitioned into s groups of N zones, one for each satellite in the cluster. Without loss of generality, we assume that zone j is visible by satellite h if and only if $(h - 1)N + 1 \leq i \leq hN$, i.e., the zones visible by the same satellite have consecutive indexes. If zone i is visible by satellite h, zone j by satellite $k(h \neq k)$, and the two satellites are not interconnected by an ISL, then $d_{ij} = d_{ji} = 0$. We denote with $D(h, k)$ the $N \times N$ submatrix of D representing the traffic between zones visible by satellite h and zones visible by satellite k. Of course, $D(h, h)$ represents traffic between zones visible by satellite h alone. When two satellites h and k are connected by an ISL, we call $D(h, k)$ *intersatellite submatrix*. We define the ith *row sum* r_i of D to be the sum of all entries in the ith row. Similarly, we define the jth *column sum* c_j of D as the sum of all entries in the jth column. We shall use the generic term *line* to refer either to a row or to a column. We also indicate with $T(h, k)$ the amount of traffic of the intersatellite submatrix $D(h, k)$, that is, the sum of all entries in $D(h, k)$.

A *time slot assignment* (or *transmission schedule*) of an ISL traffic matrix D is a decomposition of D into switching matrices: $D = S_1 + S_2 + \cdots + S_m$. A *switching matrix* S_i is an $sN \times sN$ ISL matrix with at most one nonzero entry in each line and at most one nonzero entry in each intersatellite submatrix. Each switching matrix represents the traffic that can be transmitted without conflicts and switch reconfigurations. The *length* of a switching matrix S_i is the magnitude L_i of the largest entry in S_i, and represents the number of consecutive time slots needed to transmit S_i. The *length* or *transmission time* of a schedule $S_1, S_2, \cdots, S_m$ is $L = L_1 + L_2 + \cdots + L_m$. A schedule for D is *optimal* if its length is minimum.

Lastly, a line (or intersatellite submatrix) of a switching matrix is called *exposed* if all entries in that line (submatrix, respectively) are zero, while is called *covered* if there is one nonzero entry in it. The main definitions given in this section are illustrated in Fig. 1.

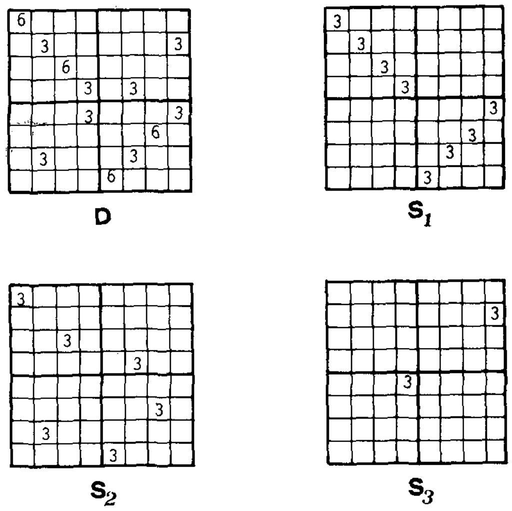


Fig. 1. An 8×8 ISL traffic matrix D for a cluster of two satellites and a length 9 schedule S_1, S_2, S_3 for it. All rows of S_2 are covered, except rows 2 and 5, which are exposed. $S_1(1, 2)$ is an exposed intersatellite submatrix, while $S_2(1, 2)$ is covered.

III. Complexity of the Problem

We now prove that it is very unlikely that any "efficient" algorithm (i.e., one running in a time bounded by a polynomial in the size of the ISL traffic matrix) can be found for determining an optimal time slot assignment when there is an arbitrary number of satellites. Indeed, the following theorem shows that this problem is NP-complete, i.e., as "hard" as a large class of problems that includes the traveling salesman problem and integer programming [28], [29]. This means that the time slot assignment problem is intrinsically intractable and can thus be solved only by "inefficient" algorithms (i.e., those running in a time which grows as an exponential function in the size of the ISL traffic matrix).

Theorem 1: The time slot assignment problem for ISL traffic matrices is NP-complete.

Proof: See the Appendix. △

The above result is rather strong, since it holds for quite restricted intersatellite link patterns and for trivial forms of traffic matrices where one may expect the problem to be much simpler. Indeed, the following corollary directly derives from the proof of the theorem.

Corollary: The time slot assignment problem is NP-complete even if:

1) each satellite covers exactly 6 zones and is connected via ISL's to exactly 3 other satellites;

2) each ISL allows transmission in only one direction;

3) the ISL traffic matrix is restricted to matrices for which the minimum number of switchings is 3;

4) the entries in the ISL traffic matrix are restricted to 0 or 1. △

The practical effect of Theorem 1 is that one is forced to abandon the search for efficient algorithms that find optimal solutions. Therefore, one can devise either efficient algorithms that provide solutions which are not necessarily always optimal but usually fairly close, or computationally inefficient algorithms (e.g., of the branch-and-bound type [29]) which provide optimal solutions. This strategy will be followed in the next section for the relevant subcase of clusters consisting of two satellites. Notice that Theorem 1 holds when there is an arbitrary number of satellites but is no more valid when this number is equal to two. Unfortunately, we were not able to come up with an NP-completeness proof for this case, although we deem that the two satellites cluster problem is likely NP-complete.

IV. Clusters of Two Satellites

In this section, we consider clusters consisting of two satellites connected by one ISL. Each satellite covers N zones and has $N + 1$ transponders.

Firstly, we derive a *lower bound* S on the duration of any schedule for the intersatellite matrix D.

Theorem 2: Any schedule for D has length not smaller than S where

$$S = \max \{ T(1, 2), T(2, 1), \max_{1 \leq i \leq 2N} \{r_i\}, \max_{1 \leq j \leq 2N} \{c_j\} \}.$$

Proof: All entries in the same line and in the same intersatellite submatrix must be transmitted sequentially to avoid conflicts. Hence, a lower bound is given by the maximum between the maximum traffic in intersatellite submatrices of D, and the maximum line sum of D. △

The above lower bound is not always achievable in an optimal schedule. As an example, any optimal schedule for the matrix D shown in Fig. 1 has length 9, while S is equal to 6.

A. Suboptimal Algorithms

We now present two fast suboptimal algorithms based upon the optimal algorithm for single-satellite systems proposed in [10] (for short, we shall henceforth refer to that algorithm as BCW).

The first heuristic, which we call modified-BCW (MBCW, for short), is designed to introduce as few changes as possible to algorithm BCW. Firstly, algorithm BCW is used to generate switching matrices with no line conflicts. Thus, only conflicts in scheduling intersatellite submatrices may arise. Successively, conflicting intersatellite traffic is eliminated from all generated switching matrices, and then scheduled in a strictly sequential way.

Algorithm MBCW

Step 1) *Application of algorithm BCW.*
Generate a schedule for D by algorithm BCW (this algorithm will add some "dummy" traffic to D [10]).

Step 2) *Elimination of conflicting intersatellite traffic.*
Let D' be a $2N \times 2N$ ISL traffic matrix initially set to zero. For each generated switching matrix S_i do the following. Subtract from S_i the dummy traffic, if any. If $k > 1$ entries of $D(1,2)$ are scheduled in S_i, drop from S_i $k-1$ of them. The dropped traffic is added to D'. Do the same for $D(2, 1)$.

Step 3) *Sequentially scheduling of conflicting intersatellite traffic.*
Generate an optimal schedule for D' by allocating the traffic in $D'(1, 2)$ and $D'(2, 1)$ sequentially.

Step 1) can be carried out in $O(N^{4.5})$ time [10]. Step 2) requires $O(N^2)$ time for each of the $O(N^2)$ switching matrices generated by algorithm BCW [10]. Finally, Step 3) takes $O(N^2)$ time. Hence, the overall running time of algorithm MBCW is $O(N^{4.5} + N^2N^2 + N^2) = O(N^{4.5})$.

Theorem 3: Algorithm MBCW generates schedules not longer than $2S$ and this bound is asymptotically achievable.

Proof: The schedule generated by algorithm MBCW can be divided into two parts: the first part consisting of the schedule before Step 3), and the second part being the schedule for D'. The first part does not require more than S time units, since algorithm BCW generate schedules whose length is equal to $\max\{\max_{1\le i\le 2N}\{r_i\}, \max_{1\le j\le 2N}\{c_j\}\}$ [10]. Moreover, an optimal schedule for D' does not require more than S time units too, since D' contains only some intersatellite traffic of D and can thus be scheduled in at most $\max\{T(1, 2), T(2, 1)\}$ time units. Therefore, the overall schedule length for D is not greater than $2S$.

This bound is asymptotically achievable. As an example, consider a $2N \times 2N$ ISL traffic matrix D having nonzero entries only in the main and secondary diagonals as follows: $d_{ii} = S(N-1)/N$ and $d_{i,2N-i+1} = S/N$, $i = 1, 2, \cdots, 2N$. We have that $T(1, 2) = T(2, 1) = S$ and all line sums are equal to S. Algorithm BCW may generate a two switching matrices schedule, in which S_1 contains all the d_{ii}'s entries and S_2 all the remaining entries. Step 2) of the MBCW algorithm leaves only two nonzero entries in S_2, putting the others in D'. This last matrix is then scheduled in $S(N-1)/N$ time units. Hence the length of the schedule generated by algorithm MBCW is $S + S(N-1)/N = (2 - 1/N)S$ which approaches $2S$ as N grows. △

A more careful allocation of the intersatellite traffic can lead to schedules shorter than those produced by algorithm MBCW. In particular, the matrix D introduced in the proof of Theorem 3 can be scheduled in S time units by forming each switching matrix as follows. First choose two intersatellite entries, one of $D(1, 2)$ and one of $D(2, 1)$, and then $2N-2$ nonconflicting entries of $D(1, 1)$ and $D(2, 2)$.

This idea is used in the second suboptimal algorithm we propose, which we call GREEDY. At any given time, a pair of intersatellite entries is selected, which yields a switching matrix with maximum number of covered lines. The resulting switching matrix is subtracted from the traffic matrix D, and the process is repeated until no traffic is left in $D(1, 2)$ and $D(2, 1)$. The traffic left over, if any, is then allocated by algorithm BCW. In the following, Q_i and S_i are $2N \times 2N$ matrices.

Algorithm GREEDY

Step 1) *Initialization.*
Set $i \leftarrow 1$, $S_i \leftarrow 0$ and $Q_i \leftarrow 0$.

Step 2) *Finding nonconflicting entries covering the maximum number of lines.*

2.1: Repeat Substeps 2.2 to 2.5 for each distinct pair of nonzero intersatellite entries, one in $D(1, 2)$ and the other in $D(2, 1)$. If no such pair exists, repeat Substeps 2.2 to 2.5 for each single nonzero entry in either ($D(1, 2)$ or $D(2,1)$.

2.2: Put the selected intersatellite entries (or entry) in Q_i.

2.3: Make a copy of $D(1, 1)$ and $D(2, 2)$ and remove from these copies the rows and columns of the selected intersatellite entries (entry).

2.4: Perform algorithm "max-min matching" [10], [30] on the so reduced copies of $D(1, 1)$ and $D(2, 2)$, thus finding a maximal set of nonconflicting entries, in which the size of the smallest entry is maximized, and put them in Q_i.

2.5: If Q_i covers more lines than S_i, then set $S_i \leftarrow Q_i$ and $Q_i \leftarrow 0$.

Step 3) *Forming a switching matrix.*
Form switching matrix S_i by truncating its entries to the value of its smallest nonzero entry. Set $D \leftarrow D - S_i$. If D contains at least one nonzero intersatellite entry, then set $i \leftarrow i+1$, $S_i \leftarrow 0$, $Q_i \leftarrow 0$, and go to Step 2).

Step 4) *Scheduling the remaining traffic.*
Schedule the traffic left in D by algorithm BCW.

$O(N^4)$ interations of Step 2) are needed to find a switching matrix. Each iteration requires $O(N^{2.5})$ time because of the max-min matching algorithm [10], [30]. The number of switching matrices generated is $O(N^2)$, since at least one nonzero entry is entirely scheduled in each switching matrix. Finally, algorithm BCW invoked in Step 4) has an $O(N^{4.5})$ running time [10]. Thus, the overall time complexity of algorithm GREEDY is $O(N^4N^{2.5}N^2 + N^{4.5}) = O(N^{8.5})$.

Theorem 4: Algorithm GREEDY generates schedules not longer than $2S$ and this bound is asymptotically achievable.

Proof: Algorithm GREEDY allocates all the intersatellite traffic during the first $\max\{T(1, 2), T(2, 1)\} \le S$ time units. After this time it behaves as algorithm BCW, thus requiring at most $\max\{\max_{1\le i\le 2N}\{r_i\}, \max_{i\le j\le 2N}\{c_j\}\} \le S$ additional time units [10]. Therefore, the overall schedule length for D cannot exceed $2S$.

This bound is asymptotically achievable. As an example, consider the 12×12 traffic matrix of Fig. 2. The schedule generated by algorithm GREEDY may have a length of $10A = (5/3)S$ while there are optimal schedules with length $6A = S$. The above example can be easily generalized to $2N \times 2N$ matrices, with arbitrary N, by replacing the $5A$ entry with an $(N-1)A$ entry and the $6A$ entry with an NA entry. It is easy to see that for such matrices any optimal schedule has length $NA = S$ while the GREEDY algorithm may generate schedules $2(N-1)A = 2S(N-1)/N$ long. Thus, the bound $2S$ is asymptotically approached as N grows. △

B. Optimal Algorithm

We now present an optimal algorithm of the *branch-and-bound type* [29] which can be set up using the lower bound of Theorem 2 and the foregoing suboptimal algorithms. The algorithm produces optimal schedules by an implicit *enumeration*. Such enumeration may generate an exponential number of switching matrices, thus requiring impractical running time, specially for large traffic matrices. However, when the traffic matrix is not large or has a particular distribution of its

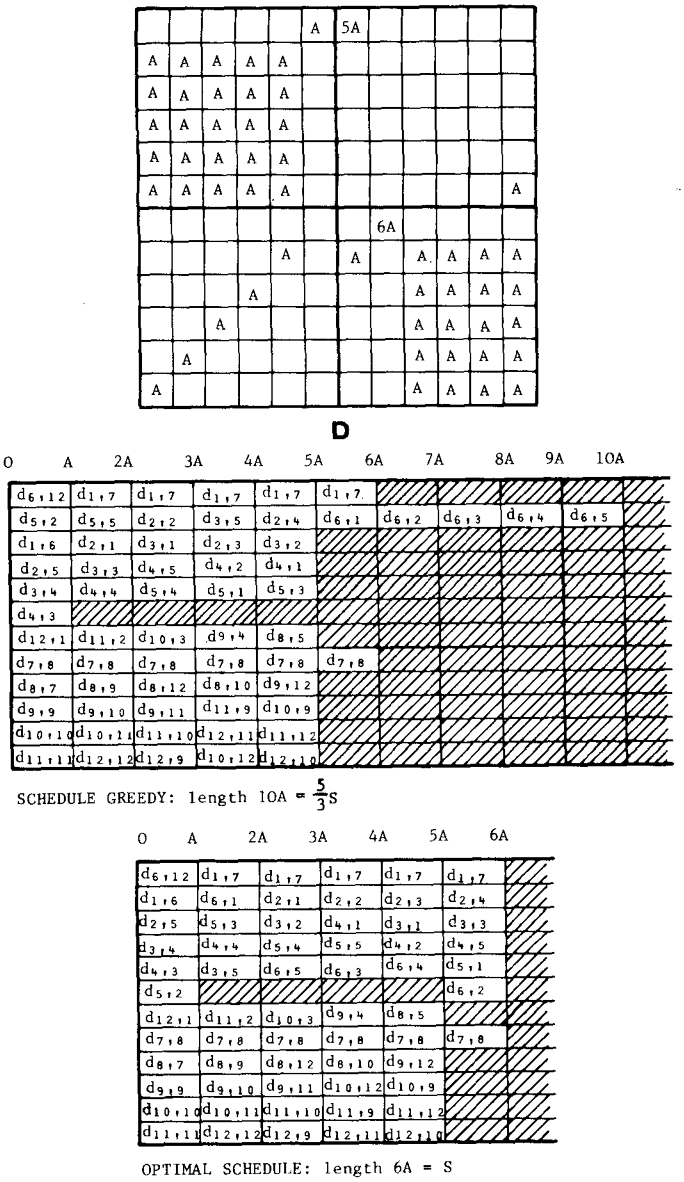


Fig. 2. Example to illustrate a bad schedule produced by the GREEDY algorithm.

nonzero entries, the algorithm may generate optimal schedules in a reasonable time.

The computation carried out by the algorithm is based upon the construction of a *tree*. Each *node* in the tree corresponds to the matrix of the remaining traffic to be allocated. The transition between a node and one of its *sons* represents the construction of a switching matrix. Each node has two parameters: *SL*, which is the length of the schedule up to that node, and LOWERBOUND, which denotes the lower bound, computed as in Theorem 2, of the traffic matrix associated to that node. At each step, the *most promising* node in the frontier of the tree, i.e., that having smallest *SL* + LOWERBOUND sum, is *expanded* by generating all possible *maximal* switching matrices of its associated traffic matrix. A maximal switching matrix is a switching matrix whose set of nonzero entries is not properly contained within a set of nonzero entries of any other switching matrix. A detailed description of the variables employed in the algorithm follows.

- *LB* denotes the lower bound of the initial traffic matrix *D*, as computed in Theorem 2.
- *UB* is the upper bound, that is, the current smallest schedule length. It can be initialized either to $2S$ or to the smallest schedule length generated by algorithms MBCW and GREEDY. Of course, if $UB = LB$ then the algorithm is not invoked.
- BESTSOLUTION contains the current shortest schedule.
- ACTIVESET is the set of all generated but not yet expanded nodes in the tree.
- $SL(D')$ is the length of the schedule that, starting from *D*, leads to the intermediate ISL traffic matrix D'.
- LOWERBOUND(D') is the lower bound on the schedule length for D', as given by Theorem 2.
- LENGTH(S') is the length of the switching matrix S'.

Algorithm OPTIMAL

Step 1) *Initialization.*
$LB \leftarrow$ LOWERBOUND(D);
$UB \leftarrow$ smallest schedule length among those generated by algorithms MBCW and GREEDY;
BESTSOLUTION $\leftarrow$ schedule with length UB;
$D_0 \leftarrow D$; ACTIVESET $\leftarrow \{D_0\}$; $SL(D_0) \leftarrow 0$.

Step 2) *Selecting the most promising node.*
If ACTIVESET is empty, then go to Step 5); else, if ACTIVESET is not empty, select matrix D_i in ACTIVESET with smallest $SL(D_i)$ + LOWERBOUND(D_i), and remove it from ACTIVESET.

Step 3) *Expanding the selected node.*
Generate all maximal switching matrices S_{ij} of D_i and set $D_{ij} \leftarrow D_i - S_{ij}$; in addition, set $SL(D_{ij}) \leftarrow SL(D_i)$ + LENGTH(S_{ij}).

Step 4) *Examining the sons of the expanded node.*
4.1: Repeat Substeps 4.2 to 4.4 for each new matrix D_{ij}. When done go to Step 2)
4.2: If $UB \leq SL(D_{ij})$ + LOWERBOUND(D_{ij}) then discard D_{ij}.
4.3: If $UB > SL(D_{ij})$ + LOWERBOUND(D_{ij}) and D_{ij} contains some nonzero entry in it, then insert D_{ij} in ACTIVESET if it is different from each matrix already in ACTIVESET, and discard it otherwise.
4.4: If D_{ij} contains only zero entries, then set $UB \leftarrow SL(D_{ij})$ and BESTSOLUTION $\leftarrow$ backtrack from D_{ij} until D_0 is reached; besides, if $UB = LB$ go to Step 5).

Step 5) *Termination.*
The optimal schedule has been found, BESTSOLUTION contains it, and *UB* is its length.

An example of the OPTIMAL algorithm is provided in Fig. 3.

C. Simulation Results

We have previously shown that algorithms MBCW and GREEDY can produce schedules as much as nearly 100 percent longer than the optimal one. In practical situations, however, traffic matrices yielding such bad schedules may be uncommon and a performance evaluation of the two suboptimal algorithms based only on a theoretical worst case bound can be misleading. Therefore, we set up simulation experiments for obtaining average schedule lengths, and thus estimate the actual performance of the proposed heuristic.

The algorithms were implemented in Pascal and run on randomly generated traffic matrices. We considered 6×6 and 8×8 ISL matrices. For each of the two matrix dimensions, we divided the simulation in subparts, depending on the value *P* of the largest entry. We chose three values of *P*, namely, 20, 50, and 100. We generated 100 matrices for each of the six subparts. The matrices were randomly generated as follows. Nearly 1/4 of the entries in submatrices $D(1, 1)$ and $D(2, 2)$ were zero, and the other entries were drawn from a uniform distribution between 0 and *P*. The intersatellite submatrices contained nearly 1/3 or 1/4 of the total traffic, depending on the dimension of the matrix. Such

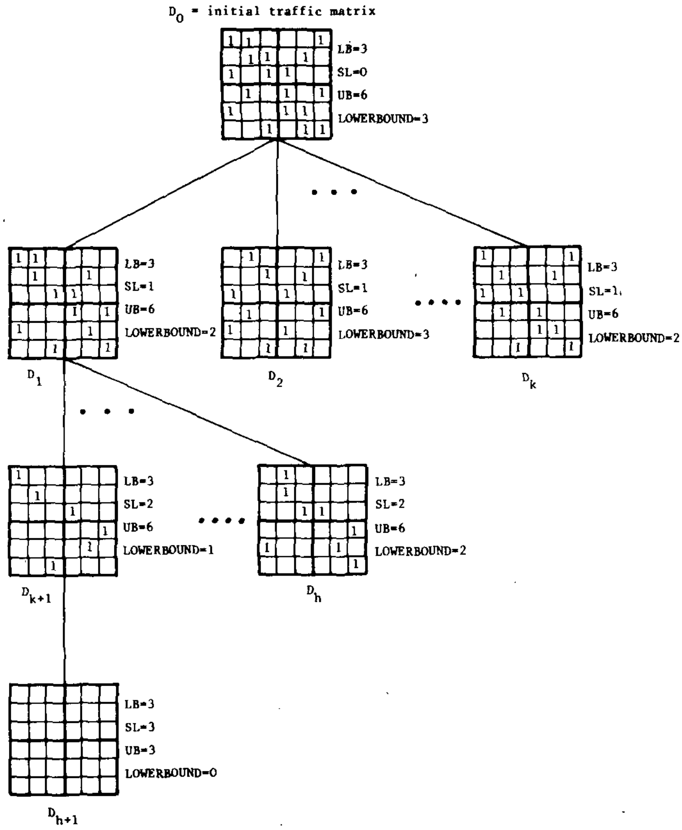


Fig. 3. Example of the OPTIMAL algorithm. The optimal schedule can be obtained by backtracking from D_{h+1}. Each switching matrix is given by the difference between father node and son node.

distribution of nonzero entries gives traffic matrices with maximum line sum close to the ISL submatrices sums, on the average. It is easy to realize that this tends to be the most unfavorable case, as the examples in Theorems 2 and 3 show. The nonzero entries in $D(1, 2)$ and $D(2, 1)$ were drawn from the same uniform distribution of the entries in $D(1, 1)$ and $D(2, 2)$. For comparison purpose, the OPTIMAL algorithm was also run on the same matrices. Tables I and II report the average schedule length for each algorithm and the average lower bound as computed in Theorem 2. Table III reports the percentage of the average surplus duration of schedules generated by the two heuristic algorithms, over the optimal schedule duration. Notice that on the average algorithm GREEDY was almost always within 1 percent of the optimal, while algorithm MBCW produced schedules not longer than 8 percent of the optimal. From our experimental trials, we observed that algorithm MBCW generated good schedules only when the intersatellite submatrices were very sparse. As soon as the intersatellite submatrices became denser, algorithm GREEDY always generated shorter schedules than MBCW. In addition, we observed that either algorithm GREEDY or MBCW produced optimal schedules in nearly 90 percent of the cases.

V. Conclusions

In this paper, we have investigated the problem of scheduling traffic for clusters of satellites interconnected via intersatellite links. We proved that the problem of finding an optimal schedule is computationally intractable for clusters including an arbitrary number of satellites, even for quite restricted intersatellite link patterns and simplified system models. We provided two fast suboptimal algorithms producing very close to optimal average schedules and a time consuming optimal algorithm for clusters including exactly two satellites. The

TABLE I
6 × 6 ISL Matrix—Average Duration of the Schedule Generated by the Algorithm

ALGORITHM	P=20	P=50	P=100
MBCW	60.44	148.27	295.96
GREEDY	58.04	142.42	284.12
OPTIMAL	57.45	141.18	280.46
LOWER BOUND	57.34	140.87	279.90

TABLE II
8 × 8 ISL Matrix—Average Duration of the Schedule Generated by the Algorithm

ALGORITHM	P=20	P=50	P=100
MBCW	78.33	184.92	381.00
GREEDY	73.10	176.58	375.78
OPTIMAL	72.67	174.82	355.67
LOWER BOUND	72.45	173.96	354.60

TABLE III
Percentage of Average Surplus Duration of the Schedule Generated by the Algorithm

ALGORITHM	MATRIX DIMENSION	P=20	P=50	P=100
MBCW	6×6	5.20 %	5.02 %	5.53 %
GREEDY	6×6	1.03 %	0.88 %	1.30 %
MBCW	8×8	7.79 %	5.78 %	7.12 %
GREEDY	8×8	0.59 %	1.01 %	0.59 %

results were given assuming that the propagation delay between the satellites in the cluster was negligibly small with respect to time slot duration. When this is not the case, our proof of intractability clearly continues to hold, but the proposed algorithms are no more correct. However, we believe that the two satellite cluster problem with nonnegligible propagation delay can be effectively handled by properly modifying the algorithms given here. This task is left as a direction for further research.

Appendix

In order to prove NP-completeness for a given problem P, one has to find an already known NP-complete problem Q which is "close" to P and to use certain transformation techniques to reduce in polynomial time Q to P so that solving Q will solve P as well. In this way, one establishes that P is at least as "hard" as Q. This implies that P is computationally intractable, since Q was already known to be computationally intractable [28], [29].

We prove that the time slot assignment problem for ISL traffic matrices is NP-complete by giving a polynomial time transformation from the known NP-complete *edge coloring* problem for cubic graphs [31].

Given: An undirected graph $G(V, E)$ such that each vertex has degree 3.

Question: Is G 3-colorable, that is, does there exist a function $f: E \rightarrow \{1, 2, 3\}$ such that $f(e) \neq f(c)$ whenever edges e and c share a common vertex?

Since this transformation is relatively laborious, we divide it into three steps. We firstly transform the graph $G(V, E)$ into a new graph $H(U, F)$ in such a way that G is 3-colorable if and only if H is 3-colorable. We then construct from H a bipartite graph $B(W, L)$ which is 3-colorable without violating certain partition constraints on its edges if and only if H is 3-colorable. We finally define from B an ISL traffic matrix D which can be scheduled into 3 time slots if and only if B is 3-colorable (and hence if and only of G is 3-colorable).

Step 1: Construction of H(U, F)

We transform the graph $G(V, E)$ into the graph $H(U, F)$ by substituting each vertex $v_i \in V$, $i = 1, \cdots, n$, with a

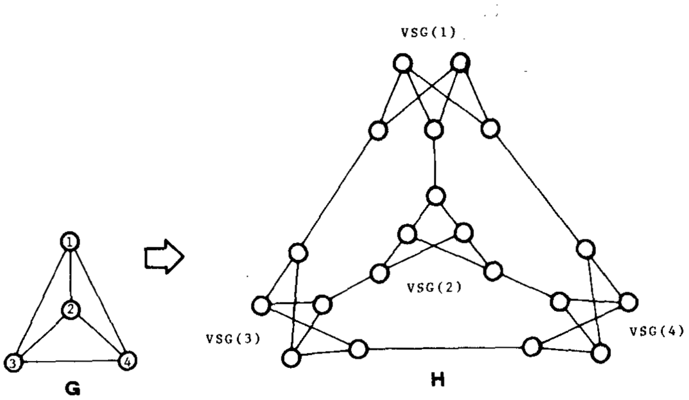


Fig. 4. Example of construction of H.

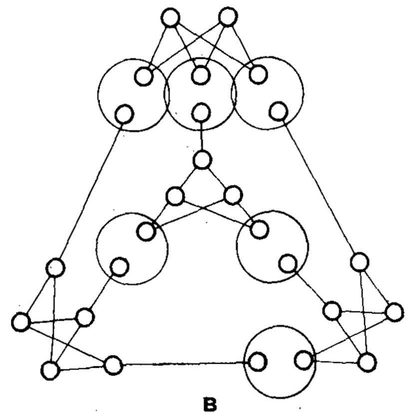


Fig. 5. Example of construction of B.

vertex substitute graph VSG. VSG is a complete bipartite graph of five vertices such that vertices 1, 2, and 3 (which we call *outlet*) are joined by an edge to the remaining two vertices (which we call *core*).

We then construct a sequence of graphs $G = G_0, G_1, \cdots, G_n = H$ as follows. To construct G_i, select vertex v_i from G_{i-1} and replace it with a copy of VSG, say VSG(i). Let the neighbors of v_i in G_{i-1} be u_1, u_2 and u_3. Replace each edge $\{u_j, v_i\}$ by an edge joining u_j to outlet j of VSG(i). An example of the above transformation is outlined in Fig. 4.

Observe that each VSG is 3-colorable, not 2-colorable, and that in any 3-coloring each outlet has its two edges joining it to the core vertices colored with a different pair of colors. This forces all the three edges joining in H the outlets of each VSG to outlets of other vertex substitute graphs to be colored with 3 different colors. Thus any 3-coloring of H can be transformed into a 3-coloring of G by coloring each edge of G with the same color of the corresponding edge of H. Conversely, any 3-coloring of G can be transformed into a 3-coloring of H by coloring each edge of H joining two outlets with the same color of the corresponding edge of G, and by coloring the edges joining each outlet in each VSG to the core vertices with the remaining two colors.

Step 2: Construction of the Bipartite Graph B(W, L)

We construct from H the graph $B(W, L)$ for the following variant of the edge coloring problem.

Given: A bipartite graph $B(W, L)$ such that each vertex in W has degree 3 or less, and a partition of the edge set L into disjoint subsets $L_1, \cdots, L_j$.

Question: Is B 3-colorable, that is, does there exist a function $g:L \rightarrow \{1, 2, 3\}$ such that $g(b) \neq g(m)$ whenever the edges b and m share a common vertex and/or belong to a common set L_i in the partition?

Initially, let $B = H$. For each edge e joining two outlets, say outlet k of VSG(i) and outlet h of VSG(j), with $i < j$, delete e from B, introduce a new vertex, and add a new edge joining the new vertex with outlet h of VSG(j). Moreover, define a set $L(i, k, j, h)$ in the partition by including the new edge and the two edges joining outlet k of VSG(i) to the core vertices. An example of the whole transformation is shown in Fig. 5 where a set in the partition having 3 edges is represented by a circle enclosing one endpoint of the edges (obviously, the partition is completed by other sets each containing exactly one of the remaining edges).

It is easy to check that the resulting graph B is bipartite and that it can be colored with 3 colors without violating the edge partition constraints if and only if H is 3-colorable.

Step 3: Construction of the ISL Traffic Matrix D

We finally construct from B the ISL traffic matrix D as follows. Let M be a 6×6 matrix with $m_{ij} = 1$, if $1 \leq i \leq 3$ and $2 \leq j \leq 3$, and $m_{ij} = 0$, otherwise. Define the $n6 \times n6$ ISL traffic matrix D in which each $D(i, i)$, $i = 1, \cdots, n$, is equal to M (and thus represents a satellite covering 6 zones). Consider each $L(i, k, j, h)$ in the edge partition of B. Swap row k of $D(i, i)$ and row $h + 3$ of $D(j, i)$. Moreover, set the entry in row h and column $k + 3$ of $D(j, i)$ to 1. An example is shown in Fig. 6. In this way, D has as many 1 entries as there are edges in B. The 1 entries in each row (or column) represent the edges incident into the same outlet (core,

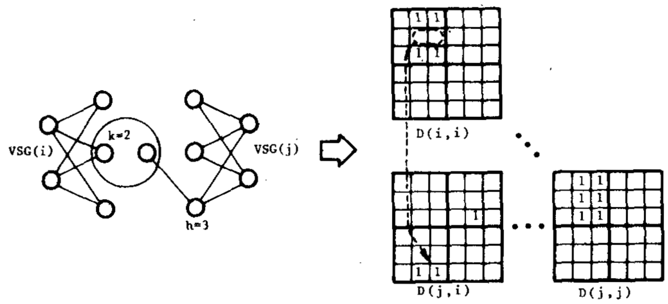


Fig. 6. Representation of the intersatellite traffic in D from satellite j to satellite $i (i < j)$.

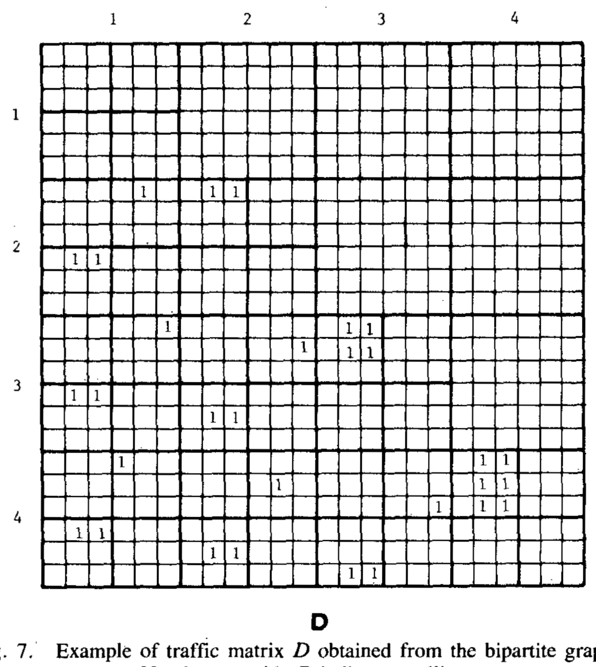


Fig. 7. Example of traffic matrix D obtained from the bipartite graph B. Numbers outside D indicate satellites.

respectively) vertex. Each intersatellite submatrix represent a set of 3 edges in the partition of B. Moreover, no additional conflict has been introduced. A complete example of the construction is exhibited in Fig. 7.

One can readily check that D can be scheduled in 3 time slots if and only if B is 3-colorable without violating the edge partition constrains, which in turn is possible if and only if G is 3-colorable. It is easy to see that all the constructions so far are possible in polynomial time. Moreover, checking whether D can be transmitted in 3 time slots is clearly in NP. This completes our proof of NP-completeness. △

REFERENCES

[1] I. Kadar, *Satellite Communications Systems.* AIAA selected reprint series, Jan. 1976.

[2] D. O. Reudink, "Spot beams promise satellite communications breakthrough," *IEEE Spectrum,* pp. 36–42, Sept. 1978.

[3] A. S. Acampora and B. R. Davis, "Efficient utilization of satellite transponders via time-division multibeam scanning," *Bell Syst. Tech. J.,* pp. 2901–2914, Oct. 1978.

[4] G. D. Dill, "TDMA, the state-of-the-art," *Rec. IEEE Electron. Aerosp. Syst. Conf.,* Sept. 1977, pp. 31.5A–31.5H.

[5] T. Scarcella and R. V. Abbott, "Orbital efficiency through satellite digital switching," *IEEE Commun. Mag.,* vol. 21, pp. 38–46, May 1983.

[6] Y. Ito, Y. Urano, T. Muratani, and M. Yamaguchi, "Analysis of a switch matrix for an SS/TDMA system," *Proc. IEEE,* vol. 65, pp. 411–419, Mar. 1977.

[7] T. Inukai, "Comments on 'Analysis of a switch matrix for an SS/TDMA system,'" *Proc. IEEE,* vol. 66, pp. 1669–1670, Dec. 1978.

[8] T. Inukai, "An efficient SS/TDMA time slot assignment algorithm," *IEEE Trans. Commun.,* vol. COM-27, pp. 1449–1455, Oct. 1979.

[9] W. W. Wu, "On the efficacy of traffic assignment in satellite-switched TDMA system," in *Proc. 4th Conf. Digital Satellite Commun.,* Oct. 1978.

[10] G. Bongiovanni, D. Coppersmith, and C. K. Wong, "An optimal time slot assignment for an SS/TDMA system with variable number of transponders," *IEEE Trans. Commun.,* vol. COM-29, pp. 721–726, May 1981.

[11] G. Bongiovanni, D. T. Tang, and C. K. Wong, "A general multibeam satellite switching algorithm," *IEEE Trans. Commun.,* vol. COM-29, pp. 1025–1036, July 1981.

[12] I. S. Gopal, D. Coppersmith, and C. K. Wong, "Minimizing packet waiting time in a multibeam satellite system," *IEEE Trans. Commun.,* vol. COM-30, pp. 305–316, Feb. 1982.

[13] I. S. Gopal, G. Bongiovanni, M. A. Bonuccelli, D. T. Tang, and C. K. Wong, "An optimal switching algorithm for multibeam satellite systems with variable bandwidth," *IEEE Trans. Commun.,* vol. COM-30, pp. 2475–2481, Nov. 1982.

[14] I. S. Gopal, "Scheduling algorithms for multibeam communication satellites," Ph.D. dissertation, Columbia Univ., New York, NY, May 1982.

[15] I. S. Gopal, M. A. Bonuccelli, and C. K. Wong, "Scheduling in multibeam satellites with interfering zones," *IEEE Trans. Commun.,* vol. COM-31, pp. 941–951, Aug. 1983.

[16] I. S. Gopal and C. K. Wong, "Minimizing the number of switchings in an SS/TDMA system," *IEEE Trans. Commun.,* vol. COM-33, pp. 497–501, June 1985.

[17] M. A. Bonuccelli, I. S. Gopal, and C. K. Wong, "Incremental time slot assignment in SS/TDMA systems," IBM T. J. Watson Res. Center, Yorktown Heights, NY, Res. Rep. RC 11377, Sept. 1985.

[18] M. Ramaswasmy and P. Dhar, "Comments on 'An efficient SS/TDMA time slot assignment algorithm,'" *IEEE Trans. Commun.,* vol. COM-32, pp. 1061–1065, Sept. 1984.

[19] G. Maral, M. Bousquet, and P. Wattier, "A practical approach to SS/TDMA time slot assignment," in *Proc. Int. Conf. Commun.,* June 1983, pp. F1.4.1–F1.4.5.

[20] D. P. Gerakoulis, T. N. Saadawi, and D. L. Schilling, "Minimizing the schedule length in an SS/TDMA system with minimum switchings," *Proc. IEEE GLOBECOM '85,* New Orleans, LA, Dec. 1985, pp. 5.7.1–5.7.5.

[21] J. L. Lewandowski, J. W. S. Liu, and C. L. Liu, "SS/TDMA time slot assignment with restricted switching modes," *IEEE Trans. Commun.,* vol. COM-31, pp. 149–154, Jan 1983.

[22] T. Inukai, "SS/TDMA networking via ISL," in *Proc. Int. Conf. Commun.,* 1981, pp. 70.6.1–70.6.5.

[23] G. Welti, "Intersatellite link for multiple access telephony," in *Proc. EASCON '78 Conf.,* 1978, pp. 432–440.

[24] D. K. Sachdev, "Satellite communications technology—challenge for the 80's," in *Proc. 8th AIAA Commun. Satellite Syst. Conf.,* 1980.

[25] D. K. Sachdev and T. Chidambaram, "Intersatellite links for international communications," in *Proc. Int. Conf. Commun.,* 1981, pp. 70.2.1–70.2.6.

[26] D. V. Z. Wadsworth, "Satellite cluster provides modular growth of communication functions," in *Proc. Int. Telemetry Conf.,* 1980.

[27] J. Deal, "Digital transmission involving intersatellite links," in *Proc. 4th Conf. Digital Satellite Commun.,* Oct. 1978.

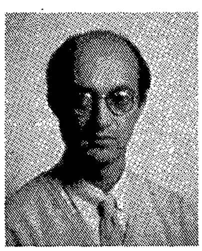
[28] M. R. Garey and D. S. Johnson, *Computers and Intractability—A Guide to the Theory of NP-Completeness.* San Francisco, CA: Freeman, 1979.

[29] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization—Algorithms and Complexity.* Englewood Cliffs, NJ: Prentice-Hall, 1982.

[30] E. L. Lawler, *Combinatorial Optimization—Networks and Matroids.* New York: Holt, 1976.

[31] I. Holyer, "The NP-completeness of edge-coloring," *SIAM J. Comput.,* vol. 10, pp. 718–720, 1981.

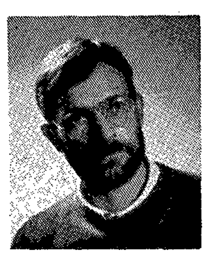
★



Alan A. Bertossi was born in London, England, on January 7, 1956. He received the Laurea degree in computer science from the University of Pisa, Pisa, Italy, in 1979.

Afterwards, he worked as a System Programmer and Designer. Since 1983 he has been associated with the Department of Computer Science, University of Pisa, Pisa, Italy, as a Research Associate and an Assistant Professor. His main research interest is the design and analysis of algorithms for combinatorial problems, even on highly parallel models of computation. He has published several articles in refereed journals on scheduling, graph, and VLSI algorithms.

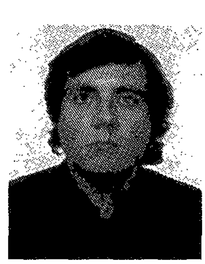
★



Giancarlo Bongiovanni received the Laurea degree in computer science from the University of Pisa, Pisa, Italy, in 1973.

In 1974 and 1975 he was with Selenia S.p.A., working on computer architectures for digital signal processing. From 1976 through 1981 he was with the Department of Computer Science, University of Pisa, Pisa, Italy, as a Research Associate and Teaching Assistant. In 1980 he was Visiting Scientist at the IBM T. J. Watson Research Center, Yorktown Heights, NY. Since 1982 he has been with the Department of Mathematics, University "La Sapienza," Rome, Italy. His present interests include algorithms for SS/TDMA systems, parallel algorithms, VLSI architectures, and computational complexity.

★



Maurizio A. Bonuccelli received the Laurea degree in computer science from the University of Pisa, Pisa, Italy, in 1975.

Since 1976 he has been associated with the Department of Computer Science, University of Pisa, Pisa, Italy, as a Research Associate and Assistant Professor. He spent 1981 at the IBM T. J. Watson Research Center, Yorktown Heights, NY, as a Visiting Scientist. His fields of interest are combinatorial optimization algorithms, with particular interest in scheduling algorithms, VLSI systems and computations, and parallel processing.