Link Scheduling in Polynomial Time

BRUCE HAJEK, SENIOR MEMBER, IEEE, AND GALEN SASAKI, MEMBER, IEEE

Abstract—**Two polynomial time algorithms are given for scheduling conversations in a spread spectrum radio network. The constraint on conversations is that each station can converse with at most one other station at a time. The first algorithm is strongly polynomial and finds a schedule of minimum length which allows each pair of neighboring stations to directly converse for a prescribed length of time. The second algorithm is designed for the situation in which messages must be relayed multiple hops. The algorithm produces, in polynomial time, a routing vector and compatible link schedule which jointly meet a prespecified end-to-end demand, such that the schedule has the smallest possible length.**

I. Introduction

A PARTICULAR model [2] for the problem of scheduling conversations between pairs of stations in a spread spectrum packet radio network will be considered. The main constraint imposed is that each station can converse with at most one other station at a time. This model is motivated by the fact that spread spectrum modulation enables multiple conversations to occur at the same time and place. In contrast, in a single-frequency narrowband system, conversations can interfere with each other even though the stations participating in different conversations are disjoint. This may quickly lead to NP-complete scheduling problems [1], so exhaustive search may be necessary [18].

Our two main problems, link scheduling to satisfy link demand and link scheduling to satisfy end-to-end demand, are described in Section II. Grotschel *et al.* [7] showed that the problem of link scheduling to satisfy link demand has polynomial time complexity (see discussion in Section V-A below). In Section III we show that the problem also has strongly polynomial time complexity by providing a strongly polynomial algorithm. To define "strongly polynomial," we define the dimension of the input as the number of input numbers plus the number of bits in the description of the combinatorial structure of the problem instance We assume the input numbers are rational and define the precision of a set of rational numbers to be the smallest integer m such that every number in the set can be expressed as the ratio of two integers, neither of which exceeds 2^m in magnitude. An algorithm is strongly polynomial if it consists of arithmetic operations (addition, multiplication, division, and comparison) and data transfers: the numbers of these operations is polynomially bounded by the dimension of the input; and the precision of the numbers appearing in the algorithm is bounded by a polynomial in the dimension and precision of the input.

In Section IV we address the problem of joint routing and scheduling to satisfy end-to-end demand using multiple hop communication. First, a polynomial time algorithm for this task which minimizes the schedule length is provided. We then show that, under a certain simplifying assumption, the routing and scheduling problems can be decoupled to a large extent, without increasing the schedule length. Final remarks are collected in Section V.

Manuscript received January 1985; revised August 1986. This work was supported by the Joint Services Electronics Program under Contract N00014-84-C-0149.

B. Hajek is with the Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois, Urbana, IL.

G. Sasaki was with the Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL. He is now with the Department of Electrical and Computer Engineering, University of Texas, Austin, TX.

IEEE Log Number 8823817.

II. Link Schedules

A. Link Schedules and Link Demand Vectors

A finite set of communication stations, henceforth called nodes, is indexed by a set N. A link is an unordered pair $[i, k]$ of nodes. Let E be the set of links $[i, k]$ such that stations i and k can carry on a conversation. The pair (N, E) is called a graph. For each $i \in N$, let $E(i)$ be the set of all links in E incident to node i (i.e., all links of the form $[i, j]$), and let $[W]$ denote the cardinality of a set W.

We say that a link is active if one of the two nodes in the link is transmitting to the other. Given a set of links M, all links in M can be simultaneously active if no two links in M have a node in common. A set M satisfying this condition is called a matching.

A schedule S is an indexed family $S = (M^\alpha, \lambda^\alpha: \alpha \in A)$, where the index set A is an arbitrary finite set, $\lambda^\alpha \geq 0$ for each α and M^α is a matching for each α. The indicator vector $I(F)$, for any subset F of E, is the vector in R^E defined by

$$I_e(F) = \begin{cases} 1, & \text{if } e \in F \\ 0, & \text{if } e \in E - F. \end{cases}$$

Using this notation, we define the length τ and the demand vector $f \in R^E$ of the schedule S by

$$\tau = \sum_\alpha \lambda^\alpha \tag{2.1}$$

and

$$f = \sum_\alpha \lambda^\alpha I(M^\alpha). \tag{2.2}$$

0018-9448/88/0900-0910$01.00

Example 1: Consider the graph (N, E) where $N = \{1,2,3,4,5\}$ and $E = \{[1,2],[2,3],[3,4],[4,5],[5,1]\}$. We will give two different schedules, S and S', such that $f_e = 1$ for all $e \in E$ for both schedules. Schedule S has the five matchings $\{1,3\},\{2,4\},\{3,5\},\{4,1\},\{5,2\}$ with corresponding values λ^α equal to 0.5 for all α. Schedule S' has the three matchings $\{1,3\},\{2,4\},\{5\}$ with corresponding values λ^α equal to 1.0 for all α. Note that S has length 2.5, while S' has length 3.

We will give three (of many) possible interpretations of a schedule $S = (M^\alpha, \lambda^\alpha: \alpha \in A)$, its length τ and demand vector f.

Interpretation 1: The two nodes of each link have some information to be transferred between them. Suppose that a typical link e must be active for f_e seconds to complete the transfer. Then all information transfers in the network can be completed during a time interval of length τ seconds as follows. The interval is partitioned into subintervals indexed by A such that the αth subinterval has duration λ^α. During the αth subinterval, links in M^α are designated to be active. Thus a given link e is active during all subintervals corresponding to α with $e \in M^\alpha$ or, equivalently, for α with $I_e(M^\alpha) = 1$. Hence, link e is active for a total of f_e seconds, as desired.

Interpretation 2: All stations have the same maximum transmission rate τ bits/s and the network is considered over a large (essentially infinite) interval of time. A typical link e will be alternately active (one of its nodes transmitting to the other at maximum rate) or not active, such that, in the long run, the nodes in link e exchange information at an average rate f_e bits/s. This is accomplished as follows. The time axis is divided into frames of duration T, where T is an arbitrary positive constant. Each frame is partitioned into a set of subframes indexed by A such that the αth subframe of each frame has length $\lambda^\alpha T/\tau$ seconds. The links in M^α are designated to be active during the αth subframe of each frame. Thus, during each frame, link e will be active for

$$\sum_\alpha I_e(M^\alpha)\lambda^\alpha T/\tau = f_e T/\tau$$

seconds, during which $f_e T$ bits can be exchanged. The long-run-average transmission rate for link e is hence f_e bit/s as desired.

Interpretation 3: Here, as in Interpretation 2, nodes transmit repeatedly over a long period of time. Suppose for a given link $e = [i,k]$ that node i needs to transmit to node k at the long-run average rate S_{ik} and node k needs to transmit to node i at the long-run average rate S_{ki}. Suppose the maximum rate of transmission from i to k (resp. k to i) is C_{ik} (resp. C_{ki}). Then link e must be active a fraction f_e of the time in the long run, where

$$f_e = \frac{S_{ik}}{C_{ik}} + \frac{S_{ki}}{C_{ki}}.$$

If the schedule length τ is less than one, then the demand can be met in the manner described in Interpretation 2, except that now the length of the αth subframe of each frame is $\lambda^\alpha T$ seconds, and an additional subframe of length $(1-\tau)T$ is included in each frame during which no links are active. Then, link e will be active for $f_e T$ seconds of each frame, which translates to being active a fraction f_e of the time in the long run.

We call any vector f in R^E with $f_e \geq 0$ for all e a link demand vector. Given such a vector f, we let $\tau(f)$ denote the smallest possible length of a schedule which satisfies demand f. In Section III we describe how to compute $\tau(f)$ and a corresponding schedule, for a prespecified link demand vector f.

Under Interpretation 1, this allows a prespecified (by f) amount of information to be exchanged in minimum time. Under Interpretation 2, this enables a prespecified set of time-average transmission rates to be achieved, using the smallest possible maximum transmission rate. Finally, under Interpretation 3, this enables a prespecified set of time-average transmission rates to be achieved subject to a prespecified set of maximum transmission rates, if it is feasible to do so.

Remark: Coffman *et al.* [4] studied a problem similar to the one studied here. Their point of view is similar to our Interpretation 1. However, they assume that the information to be transferred is grouped into files, and that once a node begins transmitting a file, it must continue to do so without interruption until the file transmission is complete. They show that, even in very special cases, the problem of minimizing the system completion time is then *NP*-complete, and they give several efficient algorithms for obtaining approximate solutions.

B. Scheduling Links to Satisfy End-to-End Demand

Typically, much of the traffic entering a packet radio network must be routed several hops. The flow of traffic in the network determines the demand on individual links, which in turn determines which link schedules are adequate. The flow of traffic is usually not uniquely determined by the end-to-end traffic requirements. In this section we address the problem of finding, for a given end-to-end demand vector, a traffic flow vector leading to the shortest length schedule.

For each i and j in N suppose that $r_i(j)$ denotes the rate (in bits/s) of the stream of traffic that enters the network at node i and which is ultimately destined for node j. We assume that each rate $r_i(j)$ is nonnegative, and that if i is equal to j then the rate is zero. The vector r is called an end-to-end demand vector. A multicommodity flow vector satisfying the end-to-end demand r is a vector u, $u = (u_{ik}(j): i, j, k \in N)$, where

$$r_i(j) + \sum_{k \in N} u_{ki}(j) - \sum_{k \in N} u_{ik}(j) = 0, \quad \text{for all } i, j \in N \text{ and } i \neq j$$

$$u_{ik}(j) \geq 0, \quad u_{ik}(i) = 0, \quad \text{for all } i, j, k \in N$$

$$u_{ik}(j) = 0, \quad \text{for all } i, j, k \in N \text{ and } [i,k] \notin E.$$

Thus $u_{ik}(j)$ denotes the traffic on link $[i,k]$, directed from i to k, and destined for node j.

The multicommodity flow vector u determines a link demand vector $f(u)$, $f(u) = (f_e(u): e \in E)$, where

$$f_{[i,k]}(u) = \sum_{j \in N} (u_{ik}(j) + u_{ki}(j)). \quad (2.3)$$

Thus, for each (undirected) link $[i, k]$ in E, $f_{[i,k]}(u)$ is the total rate of traffic flow directly between nodes i and k, summed over both directions. This definition of f in terms of u is consistent with Interpretation 2 of link demand vectors. Each $f_e(u)$ denotes a desired long-run average transmission rate, and the corresponding minimum schedule length $\tau(f(u))$ is the transmission rate (for all active links) sufficient to satisfy the end-to-end demand.

In Section IV we address the following problem. Given a graph (N, E) and an end-to-end demand vector r, find a multicommodity flow u which minimizes the minimum schedule length $\tau(f(u))$, subject to the constraint that u satisfies the specified demand r.

Example 2: Suppose that $N = \{1,2,3,4\}$ and that any pair of stations can converse. Let the end-to-end demand vector be

$$r_1(2) = r_2(3) = r_3(1) = 1, \quad \text{other } r_i(j)\text{'s zero.}$$

An obvious routing vector is given by

$$u_{12}(2) = u_{23}(3) = u_{31}(1) = 1, \quad \text{other } u_{ik}(j)\text{'s zero.}$$

which leads to $\tau(f(u)) = 3$. However, the routing vector given by

$$u_{12}(2) = u_{23}(3) = u_{34}(1) = u_{41}(1) = 1, \quad \text{other } u_i(j)\text{'s zero}$$

leads to $\tau(f(u)) = 2$. Thus even when all sources are next to their destinations, indirect routes and forwarding can lead to a smaller value of $\tau(f(u))$.

Remark: An alternative end-to-end demand problem results by using Interpretation 3 instead of Interpretation 2. We then assume that a set of maximum transmission rates $(C_{ik}: i, k \in N)$ is given. We would set

$$f_{[i,k]}(u) = \frac{1}{C_{ik}} \sum_{j \in N} u_{ik}(j) + \frac{1}{C_{ki}} \sum_{j \in N} u_{ki}(j), \quad (2.4)$$

and then the condition $\tau(f(u)) \le 1$ indicates that the end-to-end demand can be met with the prespecified set of maximum transmission rates.

III. Computation of Shortest Link Schedules in Polynomial Time

A. Link Scheduling as a Linear Programming Problem

In this section we prove the following.

Proposition 1: Given a link demand vector f, a) $\tau(f)$ can be computed using $O(|N|^5)$ computations. and b) a schedule of length $\tau(f)$ satisfying demand f can be computed using $O(|E||N|^5)$ computations.

Remark: Ogier [14] recently used a decomposition method to obtain an algorithm for finding a schedule of length $\tau(f)$ with the improved time complexity $O(|E||N|^4)$.

We begin the proof of Proposition 1 in this subsection by showing that $\tau(f)$ can be obtained by solving a linear programming problem due to Edmonds [5]. In Sections III-B and -C we complete the proof of parts a) and b) of Proposition 1, respectively. We define the degree of f at node i by

$$d(f, i) = \sum_{e \in E(i)} f_e$$

and define the maximum degree $d(f)$ of f to be the maximum of $d(f, i)$ over all i. Obviously, $d(f) \le \tau(f)$ and $d(f)$ can be strictly smaller than $\tau(f)$. For example, if N has a total of three nodes and if the link demand for each of the three pairs of nodes is one, then $d(f) = 2$ while $\tau(f) = 3$. Let Q be a set of nodes such that the number of nodes in Q is an odd integer. We call such a Q an *odd* subset of N. Let $s(Q) = (|Q|-1)/2$ and let $L(Q)$ be the set of all links $[i, j]$ such that $i, j \in Q$.

Suppose that a minimum length schedule $S = (M^\alpha, \lambda^\alpha: \alpha \in A)$ is given for a link demand vector f. Then, for any odd subset Q, using (2.1), (2.2), and the fact that any matching can contain at most $s(Q)$ links from the set $L(Q)$, we have

$$\sum_{e \in L(Q)} f_e = \sum_{e \in L(Q)} \sum_{\alpha} \lambda^\alpha I_e(M^\alpha) \le \sum_{\alpha} \lambda^\alpha s(Q) = \tau(f) s(Q).$$

If $|Q| \ge 3$ then dividing through by $s(Q)$ gives us

$$s(Q)^{-1} \sum_{e \in L(Q)} f_e \le \tau(f).$$

By defining $s(Q)^{-1} \Sigma_{e \in L(Q)} f_e = -\infty$ if $|Q| = 1$ the above inequality gives us $\tau(f) \ge Z(f)$ where

$$Z(f) = \max \left\{ s(Q)^{-1} \sum_{e \in L(Q)} f_e: Q \text{ is an odd subset of } N \right\}.$$

Both $Z(f)$ and $d(f)$ are lower bounds to $\tau(f)$.

Lemma 1 (Edmonds): For any link demand vector f, $\tau(f) = \max\{Z(f), d(f)\}$.

Proof: $\tau(f)$ is the smallest number α such that $\alpha^{-1} f$ is in the convex hull of the set $\{I(M): M \text{ is a matching of } E\}$. This convex hull is, according to Edmond's famous polytope theorem [5], [12, p. 256], equal to the set $\boldsymbol{M}$ defined by $\boldsymbol{M} = \{x \in R_+^E: d(x) \le 1 \text{ and } Z(x) \le 1\}$. It follows that

$$\begin{aligned} \tau(f) &= \min\{\alpha: d(\alpha^{-1} f) \le 1 \text{ and } Z(\alpha^{-1} f) \le 1\} \\ &= \min\{\alpha: d(f) \le \alpha \text{ and } Z(f) \le \alpha\} \\ &= \max\{d(f), Z(f)\}. \end{aligned}$$

Since the number of odd subsets Q of N grows exponentially with $|N|$, Lemma 1 by itself does not imply that $\tau(f)$ can be computed with a number of computations growing as a polynomial in $|N|$.

We leave it to the reader to deduce the following corollary from Lemma 1. We will not be using the corollary.

Corollary: a) $\tau(f) = d(f)$ if $\{e: f_e > 0\}$ is the set of links of a bipartite graph; b) $\tau(f) \leq 3d(f)/2$; c) $\tau(f) \leq d(f) + \max\{f_e: e \in E\}$.

Remark: If a demand vector f has zero-one valued coordinates then $\tau(f)$ is known as the multichromatic index, or fractional chromatic index, of the undirected graph with link set $\{e: f_e > 0\}$. Lemma 1 has been used to obtain simple (but more elaborate than those in the corollary above) bounds on $\tau(f)$ in this special case [6, sect 19], [19]. Part b) of the corollary can be deduced from Shannon's similar result for chromatic indices [17], and part c) of the corollary can be deduced from Vizing's theorem [3, p. 260].

Lemma 1 immediately yields the following expression for $\tau(f)$, which shows how $\tau(f)$ can be found by solving a linear programming problem:

$$\tau(f) = [\max\{\beta: \beta f \in \boldsymbol{M}\}]^{-1} \tag{3.1}$$

where $\boldsymbol{M}$, defined above, can be written as

$$\boldsymbol{M} = \{x \in R^E: x_e \geq 0 \text{ for all } e \in E \tag{M.1}$$

$$d(x,i) \leq 1 \text{ for all } i \in N, \tag{M.2}$$

$$\sum_{e \in L(Q)} x_e \leq s(Q) \text{ for all odd subsets } Q \text{ of } N\}. \tag{M.3}$$

The constraints of type (M.3) are called matching constraints, and as mentioned before, the number of these constraints grows exponentially with $|N|$.

B. Computation of the Minimum Schedule Length

Part a) of Proposition 1 follows immediately from (3.1) for $\tau(f)$ and the following lemma, by setting $g = 0$ and $h = f$.

Lemma 2: Let $g, h \in R^E$, where g is an indicator vector of a matching and h is not the zero vector. Then the following problem can be solved using $O(|N|^5)$ computations (we use a^T to denote the transpose of a vector a).

(P) Compute $\alpha = \max\{\beta: g + \beta h \in \boldsymbol{M}\}$ and return the vector a and scalar b, where $a^T x \leq b$ is an inequality of $\boldsymbol{M}$, $a^T(g + \alpha H) = b$ and $a^T h \neq 0$.

Remark: The hyperplane $\{x \in R: a^T x = b\}$ separates $\boldsymbol{M}$ and $g + (\epsilon + \alpha)h$ for all $\epsilon > 0$.

Proof of Lemma 2: To prove Lemma 2 we present an algorithm, called SOLVE.P, and show that it solves P in $O(|N|^5)$ computations. The algorithm first initializes the parameter $\alpha > 0$ so that $g + \alpha h \notin \boldsymbol{M}$. Whenever an inequality of $\boldsymbol{M}$ which $g + \alpha H$ violates is found, α is decreased so that $g + \alpha h$ satisfies the inequality with equality and the inequality is stored in registers a and b; i.e., $a^T x \leq b$ is the violated inequality of $\boldsymbol{M}$. Since a and b always correspond to an inequality previously violated by $g + \alpha h$, $a^T h \neq 0$.

Initialization of SOLVE.P: Find an $e \notin E$ such that $h_e \neq 0$. Such an e exists, since $h \neq 0$. Let $\alpha > 0$ be such that $\alpha h_e + g_e \notin [0,1]$.

Note that $\alpha h + g \notin \boldsymbol{M}$. The rest of SOLVE.$P$ consists of two steps. Step 1 ensures that $g + \alpha h$ satisfies the inequalities of (M.1) and (M.2), and that a and b store an inequality of $\boldsymbol{M}$ such that $a^T(g + \alpha h) = b$ and $a^T h \neq 0$.

Step 1 of SOLVE.P: For each inequality of (M.1) and (M.2), do the following. If $g + \alpha h$ does not satisfy the inequality, then store the inequality in registers a and b and reduce α so that $g + \alpha h$ satisfies the inequality with equality.

Since $|E| \leq |N|^2$, it is straightforward to show that the initialization and Step 1 take $O(|N|^2)$ time. In Step 2 of SOLVE.P we repeatedly look for a matching constraint violated by $g + \alpha h$. We then store this constraint in a and b, and decrease α so that the constraint is satisfied with equality. Since the number of matching constraints is exponential in $|N|$, we do not check each of these constraints, as we do in Step 1. Instead, we use a method of Padberg and Rao [15] which is based on a simple relationship between violated matching constraints and minimum odd cut sets.

Given a demand vector x and subsets A and B of N, we define $x(A:B)$ by

$$x(A:B) = \sum_{j \in A} \sum_{j \in B} x_{[i,j]}.$$

Using the elementary relationship

$$\sum_{i \in Q} d(y,i) = 2 \sum_{e \in L(Q)} y_e + y(Q: N - Q),$$

it is easy to see that

$$\sum_{e \in L(Q)} y_e > s(Q) \tag{3.2a}$$

if and only if

$$y(Q: N - Q) + \sum_{i \in Q} (1 - d(y,i)) < 1. \tag{3.2b}$$

Now (3.2a) means that y violates the matching constraint of $\boldsymbol{M}$ that corresponds to Q. Thus, upon complete execution of Step 2 (described next), SOLVE.P will have solved P.

Step 2 of SOLVE.P: Repeat until we stop. Let $y = g + \alpha h$. Let Q be an odd subset of N which minimizes

$$y(Q: N - Q) + \sum_{j \in Q} (1 - d(y,i)). \tag{3.3}$$

If the minimum value is greater than or equal to one, then stop. Otherwise, the inequality

$$\sum_{e \in L(Q)} y_e \leq s(Q)$$

is violated. We store this inequality in a and b and decrease α so that the inequality holds with equality.

Since the initialization and the first step of SOLVE.P take $O(|N|^2)$ computations, all that remains is to show that Step 2 takes $O(|N|^5)$ computations. Padberg and Rao show in [15, sec. 2] that the problem of finding an odd subset Q to minimize (3.3) can be solved by finding an odd cut set in a graph with node set obtained by adding

one more node to N. Padberg and Rao [15, sec. 1] also show that the odd cut set problem can be solved by a simple modification of the elegant Gomory and Hu method for finding the minimum cut sets for all pairs of nodes in a graph. The method requires at most $|N|$ iterations, where the main step in each iteration is to find a maximum flow between two nodes in the enlarged network or a shrunken version of it. Maximum flows can be found in $O(|N|^3)$ steps, so that the body of Step 2 can be executed in $O(|N|^2)$ steps.

Finally, to complete the proof, we show that the body of Step 2 is executed at most $|N|+2$ times. We use the following equivalence relation among odd subsets of N. If Q and Q' are odd subsets of N, we say they are equivalent if $\gamma_g(Q) = \gamma_g(Q')$, where

$$\gamma_g(Q) = g(Q: N-Q) + \sum_{i \in Q} (1 - d(g,i)).$$

Since g is the indicator vector of a matching, it is easy to check that $\gamma_g(\cdot)$ takes values in the set $\{0,1,\cdots,|N|\}$, and hence that there are at most $|N|+1$ equivalence classes.

Since $y = g + \alpha h$, the quantity in (3.3) is equal to

$$\gamma_g(Q) + \alpha\left[h(Q: N-Q) - \sum_{i \in Q} d(h,i)\right].$$

Since $\gamma_g(Q)$ is the same for all Q in an equivalence class, if we restrict Q to lie in a fixed equivalence class, the set of Q's minimizing (3.3) does not depend on α as long as α is positive. Thus the values of Q for distinct passes through the body of Step 2 (excluding the final pass) are in distinct equivalence classes. Since there are at most $|N|+1$ equivalence classes, the body of Step 2 is executed at most $|N|+2$ times. This completes the proof of Lemma 2.

C. Computation of the Associated Minimum Length Schedule

Finding a link schedule of length $\tau(f)$ is equivalent to finding the vertices of $\boldsymbol{M}$ such that a convex combination of these vertices equals $\tau(f)^{-1}f$. These vertices of $\boldsymbol{M}$ can be constructed using the following method used in the proof of [7, theorem 3.9]. We construct a sequence $x_0, x_1, \cdots, x_J$ of vertices, a sequence $y_0, y_1, \cdots, y_J$ of points and a (possibly null, if $J=0$) sequence $F_1, F_2, \cdots, F_J$ of facets of $\boldsymbol{M}$ as follows. Let x_0 be the indicator of any matching and let $y_0 = \tau(f)^{-1}f$. If $x_0 = y_0$ set $J=0$. Now suppose that $j \geq 0$, that if $j \geq 1$ then x_i, y_i, and F_i are defined for $1 \leq i \leq j$, and that $x_i \neq y_i$ for $i < j$. If $x_j = y_j$ set $J = j$. Otherwise, let y_{j+1} be the last point of $\boldsymbol{M}$ on the semi-line from x_j through y_j, i.e., $y_{j+1} = x_j + \alpha_j(y_j - x_j)$, where $\alpha_j = \max\{\alpha: x_j + \alpha(y_j - x_j) \in \boldsymbol{M}\}$. Let a_j and b_j be such that $a_j^T x \leq b_j$ is an inequality of $\boldsymbol{M}$, $a_j^T y_{j+1} = b_j$, and $a_j^T(y_j - x_j) \neq 0$. Let $F_{j+1} = \{x \in R^E: a_j^T x = b_j\} \cap \boldsymbol{M}$, and let x_{j+1} be a vertex of $F_1 \cap \cdots \cap F_{j+1}$. It is straightforward to prove by induction that $x_i, y_i \in F_j$ for $i \geq j$, $\tau(f)^{-1}f$ is a convex combination of $x_0, \cdots, x_j, y_j$, and the dimension of $F_1 \cap \cdots \cap F_j$ is not greater than $|E| - j$. Hence $J \leq |E|$ and $\tau(f)^{-1}f$ is contained in the convex hull of $x_0, \cdots, x_J$. The vertex x_{j+1} of $F_1 \cap \cdots \cap F_{j+1}$ can be obtained by solving the following problem

$$\max\left\{\sum_{i=0}^{j} a_i^T x: x \in \boldsymbol{M}\right\}. \tag{3.4}$$

We now show that this method takes $O(|N|^5|E|)$ computations. First note that the computation of x_0 and y_0 takes $O(|N|^5)$ computations, the number of computations it takes to compute $\tau(f)$ (see Proposition 1-a)). Given y_j and x_j, we know from Lemma 2 that α_j, b_j, and a_j can be computed in $O(|N|^5)$ computations. It takes $O(|E|)$ computations to compute y_{j+1} from x_j, y_j, and α_j. It takes $O(|N|^3)$ computations to find x_{j+1} from $\{a_i: i \leq j\}$, since the maximization of (3.4) is a maximum weighted matching problem and there is an $O(|N|^3)$ algorithm to solve this problem [12]. Thus the computation of x_j, y_j, α_j, a_j, and b_j takes $O(|N|^5)$ computations for each j. Since we compute x_j, y_j, α_j, a_j, and b_j only for $j \leq J$, the method will take $O(|E||N|^5)$ computations.

To find $\{\lambda_0, \cdots, \lambda_J\}$ such that $\sum_{i=0}^{J} \lambda_j x_j = \tau(f)^{-1}f$ we note that for $0 \leq j < J$, $y_j = (1 - \alpha_j^{-1})x_j + \alpha_j^{-1} y_{j+1}$. Thus $\lambda_j = (1 - \alpha_j^{-1})\prod_{0 \leq k < j} \alpha_k^{-1}$ for $j < J$ and $\lambda_J = \prod_{0 \leq k < J} \alpha_k^{-1}$ with the usual convention that the product of an empty set of numbers is one. It follows that $O(|E|)$ computations are needed to compute the λ_j from the α_j. Since a schedule that satisfies link demand f is $(M^j, \tau(f)\lambda_j: j \in \{0, \cdots, J\})$, where matching M^j corresponds to vertex x_j, the proof of part b) of Proposition 1 is complete.

IV. Scheduling Links to Satisfy End-to-End Demand

A. The General Case

Suppose that a graph (N, E) and an end-to-end demand vector r are given as described in Section II-B. For simplicity we will use (2.3) to define the link demand vector $f(u)$ corresponding to a multicommodity flow vector u, although Proposition 2 to follow can be modified for the more general case where (2.4) is used.

Proposition 2: It is possible to find, using a number of computations bounded by a polynomial in $|N|$ and the precision of r,

a) a multicommodity flow vector u^* that minimizes $\tau(f(u))$ subject to satisfying r, and
b) a link schedule for the link demand vector $f(u^*)$.

Proof: We only need to show how to find the flow vector u^*, since, by Proposition 1, the corresponding schedule can be found in $O(|N|^7)$ steps once u^* is found. The problem of finding an optimal flow u^* and its resulting schedule length α^*, $\alpha^* = \tau(f(u^*))$, can be written as

the linear programming problem P.

(P^*) Find (α, u) in $N_r \cap K$ with a minimum α. where

$$N_r = \{(\hat{\alpha}, \hat{u}): \hat{u} \text{ is a multicommodity flow vector satisfying } r\},$$

$$K = \{(\hat{\alpha}, \hat{u}): f(\hat{u}) \in \hat{\alpha} M\},$$

$$\hat{\alpha} M = \{\hat{\alpha} x: x \in M\}.$$

The number of linear constraints defining the set K is exponential in $|N|$. Nevertheless, if a polynomial time algorithm can be given for identifying a violated constraint of N_r or K, then the version of the ellipsoid algorithm described in [11, proof of theorem 2] can be used to solve problem P^* in polynomial time. Hence the proof of Proposition 2 will be complete once we prove the following lemma.

Lemma 3: Given a network (N, E), an end-to-end demand vector r, a multicommodity vector u, and a number α, the problem $\bar{P}$ defined below can be solved using a number of computations no larger than a fixed polynomial in $|N|$ and in the precision of r.

($\bar{P}$) If $(\alpha, u) \in N_r \cap K$, then return "yes." If not, then return a triple (z, ζ, θ) that corresponds to a linear constraint of N_r or K which is violated (i.e., either $z \cdot \alpha' + \zeta^T \cdot u' \leq \theta$ or $z \cdot \alpha' + \zeta^T \cdot u' = \theta$ is a linear constraint of $N_r \cap K$ and $z \cdot \alpha + \zeta^T u > \theta$).

Proof of Lemma 3: It takes polynomial time to check whether $(\alpha, u) \in N_r$ or to find a triple (z, ζ, θ) corresponding to a linear constraint of N_r that (α, u) violates, since there are $O(|N|^3)$ linear constraints in N_r. Thus the lemma is true in the case that (α, u) is not in N_r. Henceforth we assume that (α, u) is in N_r, so in particular $u \geq 0$ so that $f(u)$ is a valid link demand vector. Now (α, u) is in K if and only if $\tau(f(u)) \leq \alpha$. Lemma 2 guarantees that both $\tau(f(u))$ and a critical constraint from M showing that $\tau(f(u))$ is indeed the minimum length of schedules for link demand $f(u)$ can be found in polynomial time. If $\tau(f(u)) \leq \alpha$ we return "yes" for problem $\bar{P}$. Otherwise, the linear constraint (z, ζ, θ) of N_r which is violated by (α, u) can be easily obtained from the critical constraint from M computed for $f(u)$.

B. Simplication for a Special Case

We continue to consider the problem of scheduling links subject to an end-to-end demand. It will be shown that the problem simplifies under the following assumption on the network (N, E) and end-to-end demand vector r.

Assumption A: There is a subset T of N called the set of terminals such that a) $r_i(j) = 0$ unless $i, j \in T$; b) nodes in T have only one neighbor.

We call a multicommodity flow vector u loop-free if for each j, the set of ordered pairs (i, k) such that $u_{ik}(j) > 0$ does not contain any directed cycles.

Proposition 3: Suppose Assumption A is true, and suppose that u is a loop-free flow satisfying the demand r. Then the minimum schedule length implied by u is determined by the maximum degree, that is, $\tau(f(u)) = d(f(u))$.

Proof: Let Q be a subset of N with cardinality $2s+1$ for some strictly positive integer s and let L consist of all (undirected) links $[i, j]$ with i and j in Q. By Lemma 1, we only need to establish that

$$\sum_{e \in L} f_e \leq s d(f). \tag{4.1}$$

Suppose first that Q contains no terminals. We establish that

$$\sum_{e \in L} f_e \leq \frac{s}{2s+1} \sum_{i \in Q} d(f, i) \tag{4.2}$$

and this clearly implies (4.1). Consider a portion of traffic of size ϵ which is routed on a path through Q. Its contribution to the left-side of (4.2) is $k\epsilon$ where k is the number of links of the path which are contained in $L(Q)$. Since the flow is loop-free, we have $k \leq 2s$. On the other hand, its contribution to the sum on the right side of (4.2) is at least $(2k+2)\epsilon$ because the portion of traffic visits at least $k+1$ nodes of Q. Since $k/(2k+2)$ is at most equal to $s/(2s+1)$, the contribution to the left side is at most equal to $s/(2s+1)$ times its contribution to the sum in the right side of (4.1). Summing over all portions of traffic, we obtain (4.2).

For the general case, we argue by induction on the number of terminal nodes in Q. We have just established (4.1) in the case that Q contains zero terminals. Suppose, then, that $J \geq 1$ and that (4.1) is true for any odd (≥ 3) cardinality Q containing at most $J-1$ terminal nodes; consider a particular Q containing exactly J terminals.

Let i be a terminal node in Q. Let k be the neighbor of node i if the neighbor of node i is in Q. Otherwise, let k denote an arbitrary node in Q other than i. Let $Q' = Q - \{i, k\}$, and let L' consist of all unordered pairs $[u, v]$ with both u and v in Q'. Each e in L with $f_e > 0$ is either in L' or else it has k as an endpoint. Thus

$$\sum_{e \in L} f_e = d(f, k) + \sum_{e \in L'} f_e.$$

By the induction hypothesis, the sum of f_e terms on the right side of (4.3) is at most $(s-1)d(f)$, so (4.1) is true for the given Q. This completes the inductive proof.

Remarks:

1) In some situations Assumption A can be made to hold by enlarging the network model by adding terminals in a natural way.

2) Proposition 3 is to be compared with Lemma 1.

3) As noted in the corollary to Lemma 1, the conclusion of Proposition 3 is true if assumption A is replaced by the assumption that (N, E) is a bipartite graph.

4) By Proposition 3, if Assumption A is true then the problem of finding a flow vector and schedule of minimum length subject to specified end-to-end demand can be done in the following way. The first step is to find a multicommodity flow u which minimizes $d(f(u))$ among all multicommodity flows u satisfying the end-to-end demand. This

is easily expressed as a linear programming problem with $O(|N|^3)$ constraints, and it can be solved in polynomial time by the ellipsoid algorithm. Next, the resulting multicommodity flow can be changed to a loop-free multicommodity flow satisfying the same demand and having the same value of $d(f(u))$ (see [9], for example, for a polynomial time algorithm). Finally, the algorithm of Section III can be used to produce a schedule of minimum length.

V. Variations

A. Alternative Proofs of Polynomial-Time Complexity

If one is only interested in proving that a polynomial-time algorithm exists for solving the scheduling problems we have considered (without giving, as we did, a strongly polynomial time algorithm), then several alternative approaches based on the ellipsoid algorithm are apparent from [7]. As an extreme example, we can deduce from the existence of Edmond's maximum weighted matching algorithm and [7, theorem 3.8] that there is a polynomial-time algorithm for solving the following "separation" problem:

(S) If $y \in \boldsymbol{M}$, then return "yes." If not, return an inequality of $\boldsymbol{M}$ that is violated by y.

(As is apparent from our proof of Lemma 2, Padberg and Rao's method provides an $O(|N|^4)$ algorithm for solving (S).) Using the polynomial time algorithm for (S), the ellipsoid algorithm or a simple binary search technique can then be used to solve the problem (P) of Lemma 2 in polynomial time. We must then explicitly include the precision of the input in the definitions of problem size.

B. Extension to More General Sets of Conversations

An allowable set of conversations in our network at a given time is a matching. We can extend our polynomial-time scheduling results by replacing matchings by b-matchings with upper constraints, i.e., elements of

$$B = \{ x \in R^E \colon q_e \geq x_e \geq 0 \text{ for all } e \in E, \quad d(x, i) \leq b_i \text{ for all } i \in N, \text{ and } x \text{ is integer} \}$$

where the b_i and q_e are positive integers. These constraints might arise, for example, in a network where node i can participate in up to b_i conversations at a time, as long as no more than $q_{[ij]}$ conversations are with node j.

Edmonds (see [5]) showed that B is the set of vertices of $\boldsymbol{B}$, where

$$\boldsymbol{B} = \Bigg\{ x \in R^E \colon q_e \geq x_e \geq 0 \text{ for all } e \in E; \quad \sum_{e \in E(i)} x_e \leq b_i \text{ for all } i \in N; \quad \sum_{e \in L(W) \cup T} x_e \leq \Bigg(\sum_{i \in W} b_i + \sum_{e \in T} q_e - 1 \Bigg) / 2 \text{ for all subsets } T \text{ and } W \text{ such that } W \subset N, \; T \subset (W \colon N - W), \text{ and } \sum_{i \in W} b_i + \sum_{e \in T} q_e \text{ is odd} \Bigg\}$$

where $(W \colon N - W)$ is the cut set furnished by W, i.e., $(W \colon N - W) = \{[i, j] \in E \colon i \in W \text{ and } j \in N - W\}$. Padberg and Rao [15] also consider b-matchings. Let b^* be the maximum of b_i for i in N. By using results of [15] as before, we can provide $O(|N|^9 b^*)$ and $O(|N|^{11} b^*)$ algorithms, respectively, for computing $\tau(f)$ and corresponding schedules, when b-matchings are used. Algorithms with complexity polynomial in $|N|$, $\log b^*$ and the precision of the inputs can also be provided, using the methods described in Subsection V-A.

C. Some Related Problems that are NP-Complete

A problem related to finding minimum length schedules is to find an assignment of colors to the links of a graph such that no two links with a node in common are of the same color, and a minimum number of colors is used. This problem is, in fact, a special case of the scheduling problem considered in [4], and the minimum number of colors is called the chromatic index of the graph. As we already remarked in Section III, the minimum schedule length $\tau(f)$ that we study is a generalization of the *fractional chromatic index* of a graph.

Closely related to (and essentially more general than) the problems of coloring and scheduling links are the problems of coloring and scheduling nodes. The minimum number of colors needed to color the nodes of a graph so that no two adjacent nodes are of the same color is called the *chromatic number*, and the minimum schedule length needed to "schedule nodes" with demand one at each node is called the *fractional chromatic number*. While the fractional chromatic index can be computed in polynomial time, the problems of finding the chromatic index [10], the chromatic number [12], and the fractional chromatic number [7, p. 195] are NP-hard.

D. Practical Scheduling Algorithms

It is clear that faster, possibly nonminimum length, scheduling algorithms should be sought. There is some progress in that direction. Baker *et al.* [2] and Post *et al.* [16] observed good performance in simulations of simple greedy algorithms. Hajek [8] provided an $O(|N|^4)$ algorithm which generates link schedules of length at most $(3/2)d(f)$: such schedules are at most 50 percent longer than the minimum possible length. Finally, Coffman *et al.* [4] provided simple algorithms for their related scheduling problem which provide schedules at most two to three times longer than the shortest possible.

Acknowledgment

We thank M. Loui, E. Gafni, and M. Yannakakis for their help in guiding us through the literature.

References

[1] E. Arikan, "Some complexity results about packet radio networks," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 681–685, July 1984.

[2] D. J. Baker, J. Wieselthier, and A. Ephremides, "A distributed algorithm for scheduling the activation of links in a self-organizing

mobile radio network," *IEEE Int. Conf. Commun. Conf. Rec.*, Philadelphia, PA, June 1982, pp. 2F.6.1–2F.6.5.
[3] C. Berge, *Graphs and Hypergraphs*. New York: North-Holland, 1976.
[4] E. G. Coffman, Jr., M. R. Garey, D. S. Johnson, and A. S. Lapaugh, "Scheduling file transfers," *SIAM J. Computing*, to appear.
[5] J. Edmonds, "Maximum matching and a polyhedron with (0,1) vertices," *J. Res. Nat. Bur. Standards*, Sec. B, vol. 69, pp. 125–130, 1965.
[6] Fiorni and Wilson, *Edge-Colourings of Graphs*. San Francisco, CA: Pitman, 1977.
[7] M. Grotschel, L. Lovasz, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, no. 2, pp. 169–197, 1981.
[8] B. Hajek, "Link schedules, flows, and the multichromatic index of graphs," in *Proc. 1984 Conf. Information Sciences and Systems*, Mar. 1984, pp. 498–502.
[9] B. Hajek and R. G. Ogier, "Optimal dynamic routing with continuous traffic," *Networks*, vol. 14, pp. 457–487, 1984.
[10] I. Hoyler, "The NP-completeness of edge-coloring," *SIAM J. Comput.*, vol. 10, pp. 718–720, 1981.
[11] R. M. Karp and C. H. Papadimitriou, "On linear characterizations of combinatorial approximation problems," *SIAM J. Computing*, vol. 11, pp. 620–632, 1982.
[12] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rinehart & Winston, 1976.
[13] V. M. Malhotra, M. P. Kumar, and S. N. Maheshwari, "An $O(|V|^3)$ algorithm for finding maximal flows in networks," *Inform. Processing Lett.*, vol. 7, pp. 277–278, 1978.
[14] R. G. Ogier, "A decomposition method for optimal link scheduling," in *Proc. Allerton Conf. Communications, Control, and Computing*, vol. 24, pp. 822–823, 1986.
[15] M. W. Padberg and M. R. Rao, "Odd minimum cut-sets and *b*-matchings," *Math. Operations Res.*, vol. 7, no. 1, pp. 67–80, Feb. 1982.
[16] M. J. Post, P. E. Sarachik, and A. S. Kershenbaum, "A 'biased greedy' algorithm for scheduling multi-hop radio networks," in *Proc. 1985 Conf. Information Sciences and Systems*, Mar. 1985, pp. 564–572.
[17] C. E. Shannon, "A theorem on coloring the lines of a network," *J. Math. Phys.*, vol. 28, pp. 148–151, 1949.
[18] J. A. Silvester, "Perfect scheduling in multi-hop broadcast networks," in *Proc. ICCC*, London, Sept. 1982.
[19] S. Stahl, "Fractional edge coloring," *Cahiers CERO*, vol. 21, pp. 127–131.