Multicommodity Network Flows--
Computational Experience

by

Arjang A. Assad

OR 058-76 October 1976

This report was supported, in part, by the U.S. Department of Transportation under Contract DOT-TSC-1058.

ABSTRACT

This paper reports computational results with two algorithms for the linear multicommodity flow problem--a price-directive algorithm based on Dantzig-Wolfe decomposition for block-angular systems and a resource-directive algorithm based on subgradient optimization. We found the performance of the price-directive algorithm markedly superior. We then use the code to solve a multi-fleet air routing problem. The results show this to be a very efficient approach to routing problems.

I. Introduction

This paper describes the implementation of two solution techniques for linear multicommodity (henceforth abbreviated m.c.) flow problems and reports computational experience with the two algorithms. This general class of problems arise when distinct commodities are shipped from their sources to the respective destination along the arcs of an underlying capacitated network. The commodities compete for arc capacity and the objective is to minimize total routing costs subject to the capacity constraints as well as the individual flow balances for each commodity.

We aim to be very brief in our description of m.c. flow problems and solution techniques since a comprehensive survey of the literature has already been compiled in [3]. Price and resource directive decomposition are described in Sections II.1, II.3, and IV.2 of [3]. In this paper we devote Ch. II to the details of the decomposition algorithms describing Dantzig-Wolfe decomposition and a subgradient resource-directive algorithm. We also demonstrate the relation between Benders' Decomposition and the latter approach. Chapters III and IV address implementation issues and computational results on a variety of test problems. In Chapter V we describe a multifleet routing problem and modify the Dantzig-Wolfe decomposition algorithm suitably to solve it.

In order to make this report self-contained we shall now give the mathematical formulation for m.c. flow problems and provide the necessary notation as developed in [3].

Consider a directed graph $G=(N,A)$ with n nodes and m arcs on which K pairs of vertices (s^k,t^k), $k=1,\dots,k$, are specified as origin-destination pairs associated with K commodities indexed by k. The arcs of G will be denoted by (i,j) for nodes i and j along which the arc is directed; alternatively arcs will be enumerated as a=1,...,m by the index a.

For each k let f^k_{ij} = flow of commodity k on arc (i,j)

F^k = value of flow to be sent from s^k to t^k

$f_{ij} = \sum_{k=1}^{K} f^k_{ij}$ = total flow on arc (i,j).

For each node iϵN let

$$A(i) = \{j\epsilon N \mid (i,j)\epsilon A\} \quad \text{and} \quad B(i) = \{j\epsilon N \mid (j,i)\epsilon A\} .$$

A multi-commodity flow (henceforth abbreviated as m.c. flow) is a set of flows $\{f^k_{ij} \mid (i,j)\epsilon A, k=1,\ldots,K\}$ satisfying the following conservation equations and non-negativity constraints for all k:

$$\sum_{j\epsilon A(i)} f^k_{ij} - \sum_{j\epsilon B(i)} f^k_{ij} = \begin{cases} F^k & \text{if } i = s^k \\ -F^k & \text{if } i = t^k \\ 0 & \text{otherwise} \end{cases} \qquad (1.1)$$

$$f^k_{ij} \geq 0 \quad \text{for all } (i,j). \qquad (1.2)$$

Choosing an enumeration of the nodes and arcs of G, one may define the vector $f^k = (f^k_1, \ldots, f^k_m)$ with f^k_a being the flow of commodity k on arc a (a=1,...,m). Correspondingly the node-arc incidence matrix of G is set up and denoted by E. Let g^k be a column vector with entries of 1 and -1 in places corresponding to s^k t^k respectively and zeroes elsewhere. Let $b^k = F^k.g^k$. With this notation equations (1.1) and (1.2) can be written as:

$$Ef^k = F^k.g^k = b^k \tag{2.1}$$

$$f^k \geq 0 . \tag{2.2}$$

Let F^k denote the feasible region defined by (2.1) and (2.2) i.e.

$$\mathcal{F}^k = \{f^k \mid Ef^k = b^k,\ f^k \geq 0\} . \tag{2.3}$$

We call an optimization problem a m.c. flow problem if it involves the constraints $f^k \varepsilon \mathcal{F}^k$ for all k possibly in conjunction with other complicating constraints. In the linear m.c. flow problem the objective function as well as the complicating constraints are linear.

The simplest constraint of this nature imposes a capacity limitation on the total flow on each arc, i.e. requires

$$f_{ij} = \sum_{k=1}^{K} f_{ij}^k \leq d_{ij} \quad (i,j)\varepsilon A \tag{3}$$

where d_{ij} is the capacity of arc (i,j). The objective function Z has the form

$$Z = \sum_{k=1}^{K} \sum_{(i,j)\varepsilon A} c_{ij}^k f_{ij}^k = \sum_{k=1}^{K} c^k \cdot f^k \tag{4}$$

where c_{ij}^k = cost per unit flow of commodity k on arc (i,j) and c^k is the corresponding cost vector with the component $c_a^k = c_{ij}^k$ on arc a = (i,j). With the above notation the linear capacitated m.c. problem can be written as

$$\text{Min} \quad z = \sum_{k=1}^{K} c^k \cdot f^k \tag{5.1}$$

$$\text{s.t.} \quad f^k \in \mathcal{F}^k \tag{5.2}$$

$$\sum_{k=1}^{K} f^k \leq d. \tag{5.3}$$

The complicating constraints (5.3) may be easily generalized to deal with general resource constraints or conversion of flow units into capacity units by introducing matrices D^k where D^k_{ra} = amount of resource r per unit flow of commodity k on arc a. Thus a generalized resource-constraint m.c. flow problem can be defined as

$$\text{Min} \quad z = \sum_{k=1}^{K} c^k \cdot f^k \tag{6.1}$$

$$f^k \in \mathcal{F}^k \tag{6.2}$$

$$\sum_{k=1}^{K} D^k f^k \leq d \tag{6.3}$$

Obviously for $D^k = I \;\; \forall k$, we recover the original formulation (5.1-3).

This problem has a block angular structure which the detached coefficient form below makes conspicuous:

$$\begin{bmatrix} E & & & & & & \\ & E & & & & & \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & & \cdot & & \\ & & & & & E & \\ I & I & \cdot & \cdot & \cdot & I & I \end{bmatrix} \cdot \begin{bmatrix} f^1 \\ f^2 \\ \cdot \\ \cdot \\ \cdot \\ f^K \\ s \end{bmatrix} = \begin{bmatrix} b^1 \\ b^2 \\ \cdot \\ \cdot \\ \cdot \\ b^K \\ d \end{bmatrix} \tag{7}$$

This structure makes the problem suitable for the application of both price-directive and resource-directive methods. The basic idea is to do away with the complicating constraints (5.3) or (6.3) so as to decompose the m.c. problem into K single-commodity subproblems where the network structure allows efficient solution techniques. In the following two chapters we give the details of the algorithms involved.

II. DETAILS OF DECOMPOSITION ALGORITHMS FOR THE MULTICOMMODITY FLOW PROBLEM.

In this chapter we describe two algorithms for the multicommodity flow problem - price directive and resource-directive decomposition - which we implemented to obtain computational results.

1. Dantzig-Wolfe Decomposition

We have described the price-directive or Dantzig-Wolfe decomposition in the survey paper [3]. Here we review the use of Dantzig-Wolfe decomposition for the multicommodity flow problem before discussion our implementation of the algorithm.

Recall that the linear multicommodity flow problem can be formulated as

$$\text{Min} \sum_{k=1}^{K} c^k \cdot f^k \qquad (1.1)$$

$$\text{subject to} \quad f^k \varepsilon \mathcal{F}^k \qquad (1.2)$$

$$D \cdot f = \sum_{k=1}^{K} D f^k \leq d \qquad (1.3)$$

where $\mathcal{F}^k = \{f^k | Ef^k = b^k = g^k F^k, \; f^k \geq 0\}$ (2)

with F^k being the required flow value for commodity k from its origin s^k to destination t^k. The matrix D in (1.3) transforms the flow vector f to a vector Df giving the flows on capacitated arcs only. Thus if the first m_o arcs are capacitated D will be the $m_o \times m$ matrix [I,0].

Letting $Q_t^k = \{f_i^k \mid i=1,\ldots,I_t^k\}$ be the set of extreme points of $\mathcal{F}^k$ on hand at the beginning of iteration t, we may express a tentative solution to (1.1) - (1.3) as

$$f^k = \sum_{i=1}^{I_t^k} \lambda_i^k f_i^k \qquad (3.1)$$

$$\sum_i \lambda_i^k = 1, \ \lambda_i^k \geq 0. \qquad (3.2)$$

Substituting for f^k in (1.1) - (1.3) and treating the λ_i^k's as decision variables, the t^{th} "restricted master problem" is obtained:

$$\text{Min} \ \sum_{k=1}^{K} \sum_{i=1}^{I_t^k} (c^k \cdot f_i^k) \cdot \lambda_i^k \qquad (4.1)$$

$$\text{subject to} \ \sum_k \sum_i (Df_i^k) \cdot \lambda_i^k \leq d \qquad (4.2)$$

$$\sum_i \lambda_i^k = 1 \quad \forall k \qquad (4.3)$$

$$\lambda_i^k \geq 0 \quad \forall k, \forall i \qquad (4.4)$$

In the above linear program we have restricted our choice of f^k by considering only a subset Q_t^k of the extreme points of $\mathcal{F}^k$. This set, however, can be enlarged if it is profitable to consider other extreme points. Let π_t and σ_t^k be <u>optimal</u> dual variables associated with (4.2) and (4.3). The criterion for entering a new column associated with an extreme point f^k of $\mathcal{F}^k$ is that it have negative reduced cost, i.e.,

$$c^k \cdot f^k - \pi_t Df^k - \sigma_t^k < 0. \qquad (5)$$

Thus to find the most favored column (in terms of reduced cost) for entry, we solve the subproblem

$$w^k = \text{Minimum } (c^k-\pi_t D)\cdot f^k \tag{6.1}$$

$$\text{subject to} \quad Ef^k = b^k \tag{6.2}$$

$$f^k \geq 0 \tag{6.3}$$

for commodity k. The subproblem requires sending the required flow of F^k units of commodity k from s^k to t^k in such a way as to minimize the routing costs with respect to the arc costs given by $(c^k-\pi_t D)$. It may be readily solved by sending all F^k units of flow along the single minimum cost chain from s^k to t^k. Thus for each commodity k, a shortest chain problem is solved as the subproblem.

If all columns price out nonnegatively for all commodities, then the optimal solution to the current restricted master is also optimal for (1.1) - (1.3). Thus our optimality criterion is

$$\underset{k}{\text{Minimum}} \{(c^k-\pi_t D)\cdot f^k - \sigma_t^k\} \geq 0 \tag{7.1}$$

or equivalently
$$w^k \geq \sigma_t^k \quad \forall k. \tag{7.2}$$

If (7.1) is not satisfied then (5) holds for at least one k, meaning that the set $S_t = \{k \mid w^k < \sigma_t^k\}$ is nonempty. For each k in S_t, we augment the set of extreme points on hand - Q_t^k - by the extreme point generated by solving (6.1) - (6.3), i.e. $f^k_{I_t^k+1} = f^k$, to form Q^k_{t+1}. Obviously for $k \notin S_t$ no new extreme point is added to Q_t^k and $Q^k_{t+1} = Q^k_t$.

To complete our description of the algorithm we should specify the initialization procedure for the master problem. The form given for the master problem in (4.1) - (4.4) does not ensure its feasibility. Since a flow of $\lambda_i^k F^k$ units of commodity k is sent along the chain corresponding to f_i^k, it is easy to see how the capacity constraints in (4.2) would limit this flow resulting in the λ_i^k's to sum up to a value less than one. This problem is most pronounced at the initial stages where I_t^k's are small so that the number of available chains is not sufficiently large to allow us to distribute the flow so as to avoid the interference of capacity constraints. Thus it is necessary to add artificial variables λ_a^k in (4.3) to ensure feasibility. The restricted master will then read:

$$\text{Min} \quad \sum_{k=1}^{K} \sum_{i} (c^k \cdot f_i^k)\lambda_i^k + \sum_{k=1}^{K} c_a^k \cdot \lambda_a^k \tag{8.1}$$

$$\text{subject to} \quad \sum_{k} \sum_{i} (Df_i^k) \cdot \cdot \lambda^k \leq d \tag{8.2}$$

$$\sum_{k} \lambda_i^k + \lambda_a^k = 1 \tag{8.3}$$

$$\lambda_i^k, \lambda_a^k \geq 0$$

To initialize Q_t^k for t = 1 we solve (6.1) - (6.3) for each commodity with respect to the real arc costs c^k, i.e., with $\pi_o = 0$ in (6.1) to obtain f_1^k and we set $Q_1^k = \{f_1^k\}$, $I_1^k = 1$. This choice is motivated by the thought that these "real" shortest chains are, _a priori_, the most attractive candidates for carrying the flow. Consequently, they might carry a good portion of the flow in the optimal solution, so we may do well by entering them as soon as possible. (This conjecture was indeed borne out by the computational results.)

The costs c_a^k for the artificial columns are chosen to be large positive numbers so as to make these columns unattractive and drive the λ_a^k's out of the basis. To achieve this the costs c_a^k must be appreciably larger than the average chain costs $\overline{c^k \cdot f_i^k}$ (averaged over all possible chains from s^k to t^k). On the other hand as the dual variables associated with (8.1) - (8.4) will be of the same order of magnitude as the c_a^k's, one should avoid using very large values of c_a^k. In practice we have chosen c_a^k to be about 10-20 times the chain cost $c^k \cdot f_1^k$. Essentially the procedure given above is the "Big-M" method for obtaining a feasible solution. Tomlin [17] describes an alternative approach which involves a "Phase I" procedure for this problem. Our choice may be defended by the computational observation that for cases where capacity constraints (8.2) are nonbinding, we may obtain the optimal solution from the first master problem. Further remarks on this as well as other implementation issues will be made in the subsequent two chapters.

2. Resource-Directive Decomposition

Resource-directive methods (or decomposition by right-hand-side allocation) were discussed in sections II.3 and IV.2 of Part 1 [17] Here we shall describe an algorithm based on the idea of subgradient optimization as developed by Held, Wolfe, and Crowder [18]. We start by showing how the minimum cost flow problem can be put into a form amenable to subgradient optimization.

Consider again the problem:

$$v_1^* = \text{Min} \sum_{k=1}^{K} c^k \cdot f^k \tag{9.1}$$

subject to $$f^k \in \;^k \tag{9.2}$$

$$\sum_{k=1}^{K} f^k \leq d \tag{9.3}$$

The complicating constraints in (9.3) may be avoided if one knew how much of the scarce capacity resource each commodity requires in the optimal solution. The problem (9.1) - (9.3) can then be decomposed into K separate single commodity capacitated flow problems. This motivates allocating capacities $y^1,\ldots,y^K$ to the commodities and searching for an optimal allocation of d into the vectors y^k for k=1,...,K. One may thus reformulate (9.1) - (9.3) as the equivalent problem:

$$v_1^* = \text{Min} \left\{ \sum_{k=1}^{K} v_1^k(y^k) \right\} \tag{10.1}$$

subject to $$\sum_{k=1}^{K} y^k = d \tag{10.2}$$

$$y^k \geq 0 \quad \forall k \tag{10.3}$$

where

$$v_1^k(y^k) = \text{Min } c^k \cdot f^k \tag{11.1}$$

subject to $$Ef^k - g^k F_o^k = 0 \tag{11.2}$$

$$f^k \leq y^k \tag{11.3}$$

$$f^k \geq 0 \tag{11.4}$$

The single-commodity subproblem (11.1) - (11.4) minimizes routing costs given a capacity allocation y^k for commodity k. The "master problem" (10.1) - (10.3) searches over all feasible partitions of the capacity vector d into allocations $y^1,\dots,y^K$. The constraint (10.2) simply ensures that on any arc "a" the sum of capacities allocated to different commodities equals the available capacity d_a. Obviously by comparing (11.3) and (11.4) we wish to limit the capacity allocations to nonnegative values as stated in (10.3).

Note that the constraints (11.2) and (11.4) merely restate (9.2) for the required flow value F_o^k and that (11.3) describes a capacity constraint on the flow of commodity k alone. The attractiveness of resource-directive decomposition for (9.1) - (9.3) lies in the fact that each subproblem (11.1) - (11.4) is a <u>single</u> commodity minimum cost flow for any given $y^k \geq 0$. Such a problem may be solved by an efficient Out-of-Kilter algorithm [1,2,6] or by any of the recent implementations of simplex-based primal network flow codes [7,8,9]. (For a survey of the recent advances in solving minimum cost network flows, see Section III of [20]).

We now proceed to define a slightly modified but equivalent subproblem:

$$v^k(y^k) = \text{Max} \quad (c_o^k F^k - c^k f^k) \tag{12.1}$$

$$\text{subject to} \qquad E f^k - g^k F^k = 0 \tag{12.2}$$

$$f^k \leq y^k \tag{12.3}$$

$$F^k \leq F_o^k \tag{12.4}$$

$$f^k,\ F^k \geq 0 \tag{12.5}$$

Here F^k, the total flow of commodity k, is treated as a decision variable. By attaching large costs c_o^k to F^k, this variable is encouraged to assume its upper bound F_o^k, which is the required flow value in (11.2), provided the capacity constraints (12.3) do not preclude this. Thus if problem (11.1) - (11.4) is feasible, meaning that there exists a feasible flow with flow value F_o^k, then (11.1) - (11.4) and (12.1) - (12.5) have the same optimal solution $\bar{f}^k$. Moreover, in that case, at optimality we have

$$v^k(y^k) = c_o^k \bar{F}^k - c^k \bar{f}^k = c_o^k F_o^k - v_1^k(y^k). \qquad (13)$$

This manipulation shows that our two subproblem formulations are equivalent. Intuitively, (12.1) - (12.5) may be though of as first striving for maximum flow $\bar{F}^k \leq F_o^k$ through the network with arc capacities y^k; and then searching for the flow $\bar{f}^k$ with optimal routing costs among all flows with flow value $\bar{F}^k$. The advantage of (12.1) - (12.5) over (11.1) - (11.4) is that the former always possesses a feasible solution. In fact $(f^k, F^k) = (\underline{0}, 0)$ is always feasible in (12.1) - (12.5), whereas infeasibility might easily occur in (11.1) - (11.4) for a "small" capacity allocation y^k yielding a maximum flow value smaller than F_o^k. We shall later describe why we wish to avoid subproblem infeasibility.

To continue the development of the subgradient algorithm, define

$$y = (y^1, \ldots, y^K)$$

$$v(y) = \sum_{k=1}^{K} v^k(y^k) \qquad (14)$$

and $$S = \{y = (y^1, \ldots, y^K) \mid \sum_{k=1}^{K} y^k = d,\ y^k \geq 0 \ \forall k\}. \qquad (15)$$

We may restate the master problem as:

$$v^* = \text{Max} \quad v(y) \tag{16.1}$$

$$\text{subject to} \quad y \in S \tag{16.2}$$

The dual to (12.1) - (12.5) is

$$\text{Minimize} \; (\gamma^k \cdot y^k + \gamma_o^k F_o^k) \tag{17.1}$$

$$\text{subject to} \quad \pi^k E^k - \gamma^k \leq c^k \tag{17.2}$$

$$\pi^k \cdot g^k + \gamma_o^k \geq c_o^k \tag{17.3}$$

$$\gamma^k, \gamma_o^k \geq 0 \tag{17.4}$$

$$(\pi^k \text{ free})$$

where π^k, γ^k, and γ_o^k are dual variables associated with (12.2), (12.3), and (12.4) respectively. By strong duality, at optimality we have:

$$v^k(y^k) = \bar{\gamma}^k \cdot y^k + \bar{\gamma}_o^k F_o^k \tag{18}$$

so that

$$v(y) = \text{Min} \; \{ \sum_{k=1}^{K} \bar{\gamma}^k \cdot y^k + \bar{\gamma}_o^k F_o^k \} \tag{19}$$

subject to the constraints (17.2)-(17.4) for all k.

For each k the above minimization will yield an extreme point of the dual feasible region defined by (17.2) - (17.4). Thus in (19) we may minimize over the set of all extreme points of (17.2) - (17.4) for all k. Define

$$\gamma_p = (\gamma_p^1, \ldots, \gamma_p^K) \quad , \quad c_r = \sum_{k=1}^{K} \gamma_{o \cdot p}^k F_o^k \tag{20}$$

where for each k, $(\gamma_p^k, \gamma_{o \cdot p}^k)$ corresponds to an extreme point solution of (17.2) - (17.4) and the index p enumerates the finite set of all possible groupings of subproblem dual extreme points. Thus problem (19) may be stated as the finite minimization problem

$$v(y) = \text{Minimum } \{c_p + y \cdot \gamma_p \mid p=1,\ldots,n_p\}. \qquad (21)$$

The master problem (16.1) - (16.2) with the characterization of v(y) given in (21) is exactly of the form assumed by [18] for the subgradient optimization algorithm. We thus arrive at the following algorithm:

Subgradient Algorithm for the Multicommodity Flow Problem

1 - Set $i = 1$. Allocate initial arc capacities to individual commodities, that is, define the vector

$$y_o = (y_o^1, \ldots, y_o^K) \in S.$$

2 - Solve the single commodity minimum cost flow problems (12.1) - (12.5) in cycle for $k=1,\ldots,K$ using y_i^k as the right-hand-side in (12.3). Obtain dual variables $\gamma_i^k(y_i^k)$ from the optimal solution.

3 - Obtain the step size t_i and form the intermediate vector

$$\tilde{y}_i = y_i + t_i \gamma_i(y_i) \qquad (22)$$

4 - Project the intermediate vector on the convex set S to obtain the next capacity allocation vector

$$y_{i+1} = P_S(\tilde{y}_i) \qquad (23)$$

5 - Stop if termination criterion is met. Otherwise let $i \leftarrow i+1$ and go to step 2.

3. Relationship to Benders' Decomposition

We now pause to relate the derivation of (21) from (16.1) - (16.2) to Benders' algorithm as described in Sec. 7.3 of [19]. The algorithm deals with problems of the form

$$\text{Min} \quad c \cdot x + f(y) \tag{24.1}$$

$$\text{subject to} \quad Ax + F(y) \geq b \tag{24.2}$$

$$x \geq 0 \tag{24.3}$$

$$y \in S \tag{24.4}$$

where f and F are assumed to be continuous functions on the compact set S. Note that y can be thought of as the complicating variable since for given $y \in S$, (24.1) - (24.4) involves solving a linear program. This suggests that we let the flows $f^1, \ldots, f^K$ play the role of x and let y be the capacity allocation to cast (16.1) - (16.2) into the form given above. To this end, define the vectors

$$x = (f^1, \ldots, f^K, F^1, \ldots, F^K) \tag{25.1}$$

$$c = (c^1, \ldots, c^K, -c_o^1, \ldots, -c_o^K) \tag{25.2}$$

$$b_o = (0, \ldots, 0, F_o^1, \ldots, F_o^K) \tag{25.3}$$

$$b = (0, 0, b_o) \tag{25.4}$$

and block matrices:

$$E = \begin{pmatrix} E & & & -g^1 & & \\ & \ddots & & & \ddots & \\ & & E & & & -g^K \end{pmatrix} \tag{25.5}$$

$$A = \begin{pmatrix} E \\ -E \\ -I \end{pmatrix} \quad ; \qquad B = \begin{pmatrix} 0 \\ 0 \\ I \end{pmatrix} \quad ; \tag{25.4}$$

and further specify the functions in (24.1) and (24.2) as

$$f(y) = 0 \quad , \quad F(y) = By \quad \forall y \tag{25.5}$$

where, as before, y is the vector $(y^1,\ldots,y^K)$.

With the above definitions we note that

$$c\cdot x + f(y) = c\cdot x = \sum_{k=1}^{K} c^k \cdot f^k - c_o^k F^k \ , \tag{26}$$

and that (24.2) now expresses the constraints (12.2) - (12.4) <u>for all k</u>. This establishes the relation between the general form in (241.1) - (24.4) and the multicommodity flow problem of (16.1) - (16.2) with the definitions (14) and (12.1). Since (24.1) minimizes the expression in (26) which is the negative of v(y) for given y, the problems are equivalent.

Continuing with the general description of Benders' algorithm, define the cone

$$\mathcal{C} = \{u \mid uA \leq 0, \ u \geq 0\} \tag{27.1}$$

and the polyhedron

$$\mathcal{P} = \{u \mid uA \leq C, \ u \geq 0\} \tag{27.2}$$

where u is the dual variable associated with (24.2). Let u_i^r be the generators of $\mathcal{C}$ for $i=1,\ldots,n_r$ and u_p be the extreme points of $\mathcal{P}$ for $p=1,\ldots,n_p$. The master problem for Benders' algorithm is then stated as

$$\text{Minimize} \quad z \tag{28.1}$$

$$\text{subject to} \quad z \geq f(y) + u_p(b-F(y)) \qquad p=1,\ldots,n_p \tag{28.2}$$

$$u_i^r(b-F(y)) \leq 0 \qquad i=1,\ldots,n_r \tag{28.3}$$

$$y \in S \tag{28.4}$$

We focus on the constraints (28.4) - (28.4). By using Farkas' Lemma, we may show that these constraints specify exactly the set R of all y leading to feasible subproblems. More precisely

$$R = \{y \in S \mid \exists x \geq 0 \text{ such that } Ax \geq b-F(y)\}$$

$$= \{y \in S \mid y \text{ satisfies } (28.3)\} \quad . \tag{29.1}$$

The generators u_i^r are not available a priori but are obtained from unbounded duals of infeasible subproblems whenever a choice $y \notin R$ is made.

In our case, however,

$$R = \{y \in S \mid (12.1)-(12.5) \text{ has a feasible solution } \forall k\} \tag{29.2}$$

and since our subproblems were chosen to be feasible for any choice $y \geq 0$, we have

$$R = S \tag{29.3}$$

so that we can ignore the constraints in (28.3) and retain only (28.4). Upon doing so, (28.1) - (28.4) may be written as

$$\text{Min Max} \quad \{-\sum_{k=1}^{K} (\gamma_p^k y^k + \gamma_{o \cdot p}^k F_o^k) \mid p=1,\ldots,n_p\}$$

$$\text{subject to} \quad y \in S \tag{31}$$

which is equivalent to (16.1) - (16.2) once we take account of the minus sign within the brackets in (31).

This derivation shows that our problem formulation for subgradient optimization can be thought of as setting up the master problem for Benders' decomposition which can then be solved by the algorithm outlined in steps 1-5 above. This should also justify our reformulation of the subproblems to avoid infeasibility: Had we started out by applying Benders' to the original problem (9.1) - (9.3), the subproblems could have been infeasible for certain choices $y \in S$, implying that R would have been a proper subset of S; specifying the set R would require constraints of the form (28.3) which would complicate solving the master problem. This complication arises in most right-hand-side allocation methods, such as the work of Grinold [22] and the tangential approximation algorithm of Geoffrion [21] and Chap. 9 of [19], a good part of the effort goes into specifying the set R by generating approximating constraints. In our present formulation, however, solving the master problem involves no more than the projection (23).

We conclude this chapter with brief comments about the form of step sizes t_i and the mechanism of performing the projection in step 4 serving as a background to our discussion of computational issues in the subsequent chapters.

Held, Wolfe, and Crowder [18] suggest using a step size of the form

$$t_i = \lambda_i \frac{\hat{v}-v(y_i)}{||\gamma_i(y_i)||^2} \qquad (32)$$

with $\quad 0 < \epsilon < \lambda_i \leq 2$

where $\hat{v}$ is an underestimate of the optimal value v^* (i.e. $\hat{v} < v^*$), $v(y_i)$ is the optimal value of the master at iteration i with right-hand-side y_i as in (14); and finally the denominator denotes the Euclidean norm of the dual vector $\gamma_i(y_i)$ corresponding to y_i, that is,

$$||\gamma_i(y_i)||^2 = \sum_{k=1}^{K} ||\gamma^k(y_i)||^2 = \sum_k \sum_{a \varepsilon A} (\gamma_a^k(y_i^k))^2. \qquad (33)$$

Held and Karp [4] and Oettli [5] give theoretical justification for the form in (32) for $\lambda_i = 1$.

The subgradient approach is made attractive by the ease with which the projection in (23) can be performed: For each arc a in the network, the intermediate capacities $(\tilde{y}_a^1,\ldots,\tilde{y}_a^K)$ obtained from (22) have to be projected upon the set

$$S_a = \{x = (x^1,\ldots,x^K) \varepsilon \mathbb{R}^k \mid \sum_{k=1}^{K} x^k = d_a,\ x^k \geq 0\ \forall k\} \qquad (34)$$

where d_a is the total capacity of arc a. Specifically, we require the solution $y_a = (y_a^1,\ldots,y_a^K)$ to the minimum norm problem

$$\text{Minimize} \quad ||\tilde{y}_a - x||^2 \qquad (35)$$

$$\text{subject to} \quad x \varepsilon S_a$$

The procedure involves ordering the components of $\tilde{y}_a$ such that

$$y^1 \leq y^2 \leq \ldots \leq y^K.$$

III. IMPLEMENTATION AND COMPUTATIONAL RESULTS

A - THE PRICE-DIRECTIVE DECOMPOSITION ALGORITHM

In this chapter we discuss the implementation and program structure of the Dantzig-Wolfe decomposition algorithm and present computational experience obtained for that algorithm.

1. Program Structure

The program for the price-directive decomposition algorithm involves a linear programming code to solve the mater problem and shortest chain algorithm to solve the subproblems. The main program, called DCMP for decomposition, deals with the master problem and communicates with two subroutines GENCOL and BELL that are described below.

The SEXOP package (Subroutines for Experimental Optimization) developed by R. E. Marsten [14] is used to solve the master problem at each iteration. SEXOP consists of a collection of subroutines which are well suited for column generation and one designed for interactive use. In particular the number of columns in the linear program are allowed to vary from one iteration to the next. A subroutine ADDCOL can be used to add columns to the master problem. The optimal solution to the master at iteration i serves as a starting solution for iteration i+1 in SEXOP and the convexity constraints (II.8.3) are treated as GUB constraints. Thus SEXOP may be expected to solve the mater problem efficiently at each iteration. The only disadvantage of SEXOP is that the number of regular constraints (II.8.2), which correspond to the number of capacitated arcs in the network in our formulation, are restricted to be less than or equal to 99 in the current working version. It would not be

The optimal solution y_a to (35) then satisfies

$$y_a^k = \text{Max}\,\{Y^k-\lambda^*,0\} \tag{36}$$

where

$$\lambda^* = (\sum_{k=J+1}^{K} Y^k-d_a)/K-J \tag{37}$$

with J defined as

$$J = \text{Max}\,\{j \mid (\sum_{k=j+1}^{K} Y^k-d_a)/K-j > Y_j\} \tag{38}$$

The projection given above need not be performed if the capacities on arc a have not been changed in step 3 of the algorithm. Indeed, the set of arcs for which the capacities have been altered in (22) is given by

$$T_i = \{a\varepsilon A \mid \gamma_{i,a}^k(y_i) > 0 \text{ for some } k\}. \tag{39}$$

The arcs in T_i are flagged for projection. Similarly the summation over A in (33) can be limited to T_i.

Computationally one expects T_i to be small in comparison with A. In particular any arc which is not individually saturated by any commodity (i.e. for which $f_a^k < y_{i,a}^k\ \forall k$) will belong to the set $A-T_i$ and so its capacity allocation will not be altered at iteration i. This computational fact suggests that the optimal solution and dual variables of subproblem (12.1) - (12.5) at iteration i will serve as a good starting solution for the primal-dual algorithm at iteration i+1 since relatively few components of y_{i+1}^k are different from y_i^k.

advisable to use SEXOP for problems with more than around a hundred regular constraints anyway, since the program would be inefficient compared to commercial codes especially if the problem is sparse. We reiterate that a salient advantage of SEXOP is the ease of interacting with it.

The shortest chains are found be Bellman's algorithm for networks with nonnegative arc costs. The version we use, called BELL, was coded by B. Golden and is described in [10]. The codes uses a "Forward Star" network representation scheme [11] in which the arcs (i,j) A are ordered lexicographically in increasing order. Thus for a given node $i \in N$ we list all arcs $(i,j) \in A$ in the order of increasing j and then go on to node i+1 to do the same and continue until i = NN, where NN denotes the number of nodes in the network. Thus two arrays are sufficient to represent the network topology: A pointer of length NN giving the arc number (in the above ordering) of the first arc starting with node i; and an end-node array of length NA listing the j's for all $(i,j) \in A$. (NA is the number of arcs in the network). This representation scheme was extended to the rest of our code.

The subroutine BELL finds the shortest distances from a specified origin s to all other nodes in the network. The labeling assigns each node i the cost of the shortest path (s) from s to i and a predecessor node j. The output of BELL is thus a predecessor vector and a cost vector. Given a destination t one may immediately obtain the length of the shortest path from s to t from the t-entry of the cost vector and trace through the predecessor vector (starting at t) to obtain the optimal path in terms of a sequence of nodes $[s,i_1,i_2,\ldots,i_{n-1},t]$.

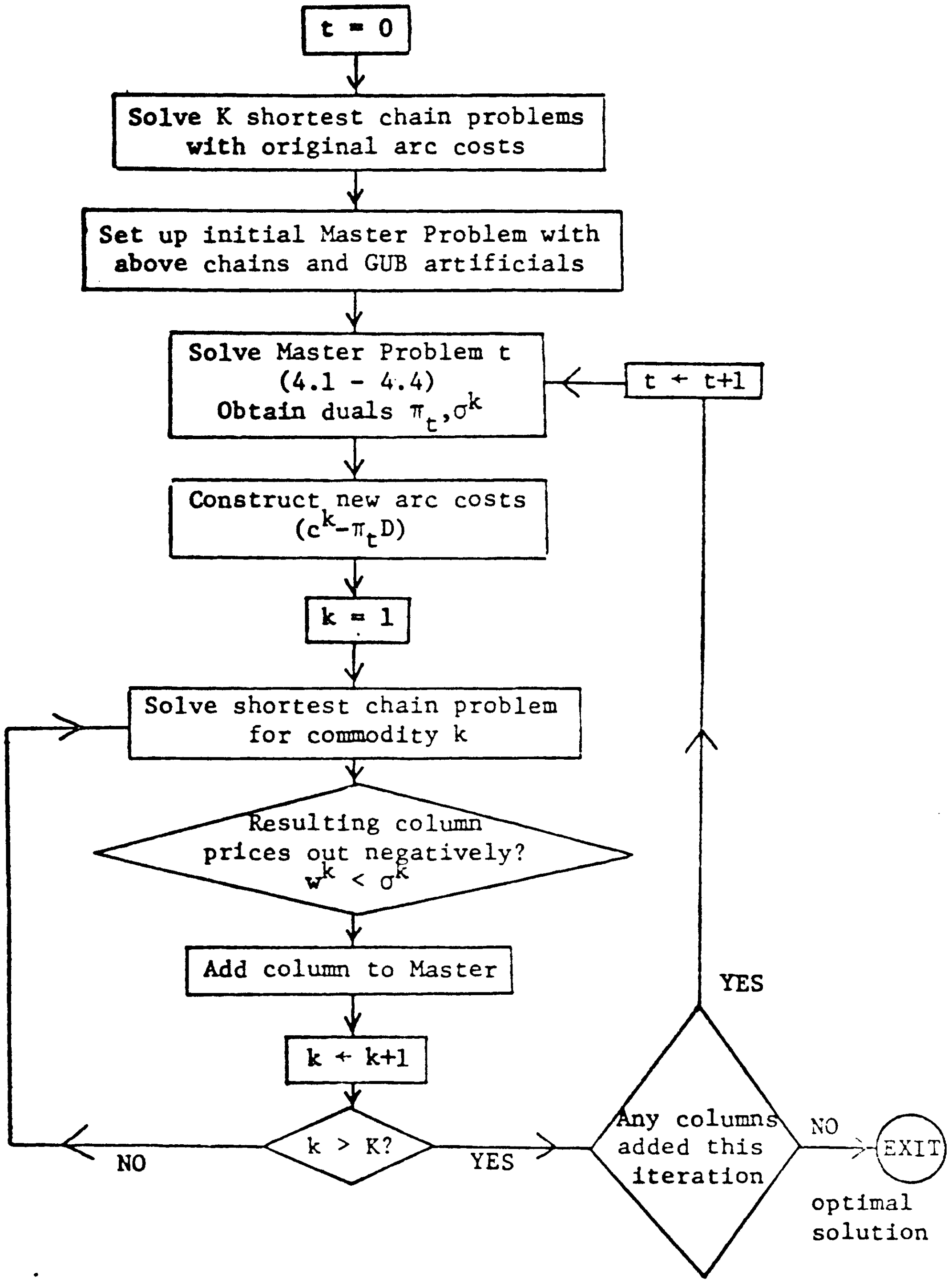
This procedure can easily be extended to a commodity with more than one destination. Given the source and n^k destinations, we call BELL once and trace through the predecessor vector n^k times to find the shortest paths from s to each of the n^k destinations.

The above is actually done in subroutine GENCOL which generates the column f^k to be added to the master problem in a form specified by ADDCOL. This column has entries equal to F^k in the places corresponding to capacitated arcs on the path from s^k to t^k and zeroes elsewhere. More specifically if an arc (i_e, i_{e+1}) on the path from s^k to t^k is capacitated and corresponds to the q^{th} capacity constraint we put F^k in the q^{th} entry of f^k. While tracing through the predecessor vector, GENCOL also finds the "real" chain costs, $c^k \cdot f^k$ which are required by SEXOP as the corresponding column costs.

The interaction between the subprograms is shown in Figure 6. The dual variables π_t from the master problem are passed on to GENCOL by DCMP. GENCOL uses π_t to obtain the modified arc costs $(c^k - \pi_t D)$ which are then passed on to BELL with the specified origin s^k. The length of the chain from s^k to t^k with respect to modified arc costs, which is w^k in (II.6.1), is then compared to the dual variable σ_t^k to decide upon entry. If the column entry criterion is satisfied, GENCOL proceeds to generate the column f^k as described above, if not control is transferred back to DCMP which then considers the next commodity k+1. If no columns are added to the master problem, it is declared to be optimal by virtue of (II.7.1) and the algorithm halts.

Figure 1

Flow-Chart for Dantzig-Wolfe Decomposition



$t = 0$
Solve K shortest chain problems with original arc costs
Set up initial Master Problem with above chains and GUB artificials
Solve Master Problem t (4.1 - 4.4) Obtain duals π_t, σ^k
$t \leftarrow t+1$
Construct new arc costs $(c^k - \pi_t D)$
$k = 1$
Solve shortest chain problem for commodity k
Resulting column prices out negatively? $w^k < \sigma^k$
Add column to Master
$k \leftarrow k+1$
$k > K$?
NO
YES
Any columns added this iteration
YES
NO
EXIT
optimal solution

2. Description of Test Problems

We shall now describe the test problems used for the computational studies. We constructed four networks to test the two decomposition codes. These are referred to as problems P1 to P4, the basic parameters of which are given in Table 1. The notation for problem parameters is as follows:

NA	=	number of arcs in the network.
NN	=	number of nodes in the network.
NCAP	=	number of capacitated arcs.
NCOM	=	number of commodities.

The basic factors which are varied in the test problems are network topology and the ratio of capacitated arcs to toal number of arcs, i.e. NCAP/NA.

As noted earlier, the current implementation of SEXOP limit NCAP to be less than 100, which explains why the maximum number of capacitated arcs in our test problems is 98. In Problems P1 and P2 all arcs are capacitated, whereas in P3 and P4 the ratio NCAP/NA is roughly .48 and .38 respectively.

The network topologies were chosen with a view twoards the modeling applications of the multicommodity flow problem: The network structure in problems P1 and P3 is essentially acyclic (with only a very few cycles) and may be expected to arise in applications to multicommodity multiperiod, production-distribution problems or more generally in the construction of time space diagrams for scheudling vehicles on a network [12]. In our examples we conceive of certain arcs as denoting passage of time in a fixed spatial location whereas the others denote spatial transportation from one point to another. Thus in a production-distribution example the "time-like" arcs represent keeping an time in inventory for one time period thereby incurring an "arc" holding cost. The "space-like" arcs, however, stand for shipment of the items in the distribution network and the arc costs for these shipments could reflect transportation costs.

The other network topology - corresponding to P2 and P4 - corresponds to an undirected communications network. Thus for nodes i and j in the network $(i,j)\varepsilon A$ implies $(j,i)\varepsilon A$. Such networks would arise naturally in intercity transportation problems as well as in the currently expanding studies of computer networks. We note that the degree of nodes in this structure is greater than in the above and the networks contain more chains. The two network structures are different enough to warrant experimentation. We note that our choice of these structures was to some extent influenced by the work of Swoveland [24] whose test problems have the above two structures. All data was generated manually, and the arc costs basically satisfy the triangle inequality (with a few exceptions in any network).

3. Computational Experience with Dantzig-Wolfe Decomposition

For each of the test problems, P1 through P4, a series of runs was made to investigate the effects of

a) the number of commodities NCOM

b) the total flow FTOT of all commodities

on the number of iterations of the decomposition algorithm (i.e. the number of master problems solved to reach optimality) and the total time required for the problem solution (excluding input-output). The total flow FTOT is defined to be the run of the fixed flow requirements F_o^k over all commodities, i.e.

$$FTOT = \sum_{k=1}^{K} F_o^k$$

That NCOM is an essential variable for experimentation deserves little explanation. Obviously in the decomposition code, this parameter determines the number of subproblems to be solved, the number of GUB constraints in the master problems, as well as the rate of growth of the columns in the master problem from one iteration to another. The role of FTOT may be explained as follows: Clearly the complicating mutual capacity constraints (II.8.2) interfere to the extent that commodities compete for capacity. If FTOT is small compared to the average arc capacity the flow of each commodity may be sent on single chain unimpeded by the capacity constraint whereas if FTOT is large, the flow of a commodity k must be apportioned over several chains. Thus the total load on the network is expected to affect solution times.

In order to have some control on the interaction between the effects of varying NCOM and FTOT; the runs for each problem were divided into to groups:

a) Holding the total flow fixed, increase the number of commodities.

In this group of runs one would start with a few commodities (i.e. OD pairs) and then successively break down the flow requirement for each commodity into flow requirements for a number of new commodities. Such runs were motivated by the idea of aggregation of commodities: A commodity k (which is specified completely by its origin-destination-OD-pair (s^k, t^k)) can be broken up into a number of "more detailed"commodities with OD pairs (s_i^k, t_i^k) $i=1,\ldots,n_k$ each with a required flow value of F_i^k subject to the restriction

$$F_o^k = \sum_{i=1}^{n_k} F_i^k$$

where F_o^k is the required flow for commodity k. Obviously FTOT remains constant in such a process of disaggregation and the load pattern on the network does not change much either since care will be taken to choose the s_i^k's and t_i^k's from the "vicinity" of s^k and t^k respectively.

b) Increase the number of commodities increasing the total flow proportionally.

Here a basic "per commodity" required flow value FAV is chosen and will be the same for all commodities k. Thus it is clear that we will have

$$F_o^k = \text{FAV} \quad \forall k, \qquad \text{and} \qquad \text{FTOT} = \text{NCOM} * \text{FAV}.$$

In this case the network traffic is becoming more congested as new commodities (OD pairs) are added.

We shall refer to these two strategies as a) and b) when discussing the computational results below.

Tables 2-5 summarize the results of our experimentation. All runs were made on the IBM 370/168 machine of MIT Information Processing Center.

The runs P1.1, P1.2, and P1.3 follow the abovementioned strategy a) with a fixed value of total flow equal to 28.0. The same strategy is used in runs P1.4 through P1.6 with a total flow of 30.0. An interesting pattern presents itself in both cases. The number of iterations (no. of master problems solved to reach optimality) - denoted ITOPT - decreases as the number of commodities increase for a fixed value of total flow. This change is due to the fact that the number of chains generated at each iteration is equal to the number of commodities. Thus many more chains are available at the end of a given iteration for a run with large NCOM compared to small NCOM. Moreover the average flow per commodity decreases in strategy a) as the number of commodities increases (F^k equals FTOT/NCOM on the average). Since each column in the master problem represents a chain with all the flow F^k assigned to it, capacity constraints are less likely to interfere when F^k grows smaller in comparison with the chain capacity.

The run P1.7 was included to test the effect of uneven loading on the network. We feel that a problem will require more iterations if the flow requirements are distributed in an uneven fashion compared to a problem with even distribution of the same total flow value. In P1.8 we consider three OD pairs with flow requirements 9,10, and 9 respectively. In P1.7 we change the requirements to 14, 12, and 2. The total flow is the same for both runs. We see that the number of iterations as well as the solution time increases substantially.

Table 3 presents the results of DCMP on P2. It may be recalled that the structure of P2 corresponds to an undirected graph. In the interests of symmetry we chose our OD pairs symmetrically as well. This means after assigning an OD pair (s^k,t^k) to an odd-numbered commodity k, commodity k+1 is given the OD pair (t^k,s^k) with the same flow value F_o^k.

The series of runs P2.5-P2.8 follow strategy b). The average per commodity flow value chosen is FAV = 6.0. Since the total flow value in runs P2.5 and P2.6 is small compared to the network capacity, the first master problem is found to be optimal. This shows the advatange of starting out with good chains in conjunction with a "Big-M method". As the number of commodities, and hence the total flow, increases, more iterations are required.

Comparing runs P2.3 and P2.4 which share the same value of total flow, once again we see the effect of increasing NCOM in terms of decreasing the number of iterations. The runs for P2 show that ITOPT is not necessarily a good indicator of the solution time required by the problem. For example P2.3 has a lower ITOPT than P2.2 but requires a much greater solution time. This, by the way, should be attributed to the SEXOP routine solving the master problem, which seems to slow down when NCOM is large (thereby having a large number of columns in each master as well as a greater number of GUB constraints); and not to the shortest chain routine solving the subproblems as we shall see.

In the runs for P3 strategies a) and b) were used in groups P3.1-P3.4 and P3.5-P3.7 respectively. Again increasing the number of commodities with fixed total flow decreases ITOPT but the solution time tends to increase nevertheless. For the second group it can be seen that both ITOPT and the solution time increase with the number of commodities.

Finally for P4, strategy b) was employed. We remark that the capacity constraints are rather sparse in this problem, in the sense that there is a good chance of being able to find an uncapacitated chain between an arbitrary pair of nodes. P4 corresponds to an undirected graph as well. A per commodity flow value of 15.00 (=FAV) was used. We note a steady increase in solution times with the number of commodities.

In empirical timing of decomposition algorithms with a master problem interacting with subproblems possessing special structure, it is a good idea to investigate how time-consuming the subproblems are to solve. In addition to locating the portion of the algorithm where most of the effort is expended; it throws some light on the effect of varying the number of subproblems - NCOM in our case. For our Dantzig-Wolfe Decomposition code, the subroutine BELL used to solve the shortest chain problems should be timed. To this end the network and cost structures of P1 through P4 were used to obtain an average time per shortest-chain calculation for the three structures. The results are shown below:

Problem Name	Number of Arcs	Number of Nodes	Ave. Time (in 10^{-3} sec.)
P1	98	47	1.47
P2	96	25	1.25
P3	204	83	2.89
P4	268	44	3.03

Thus for an average network of 200 arcs and 10 commodities which may require 10 iterations on the master level, we may expect an upper limit of .3 sec. on the total time expended in solving the shortest chain subproblems. Given that the total time for the solution of such a problem may be 2-3 sec.; this is not a substantially time-consuming activity of the code. As a further illustration of the use of this information we compare the runs P2.2 and P2.3. The number of shortest path problems solved in each run is (ITOPT+1)•NCOM•T where T is the average time per shortest chain calculation. Thus in going from P2.2 to P2.3 a total time increase of 60T = 0.075 seconds may be attributed to the additional time spent in solving shortest path problems. This shows that increasing NCOM affects the LP solution time much more than the subproblems.

We wish to conclude by noting that our price-directive decomposition code has not yet been subjected to a second level effort aimed at shortening its running time. We can easily visualize detailed portions of the code to which such an effort could be applied. In particular the subroutine GENCOL could be much more efficient in using the inputs from BELL to generate columns. Also, one may experiment with different strategies for column generation. It may be advantageous to have added columns replace some other nonbasic columns in the master problem. Another possibility is to ignore a subproblem which has priced optimally at a certain iteration for a number of subsequent iterations since the set $\bar{S}_t$ of subproblems optimal at iteration t, does not change much from one iteration to the next.

Figure 2

Interaction between DCMP and its subprograms

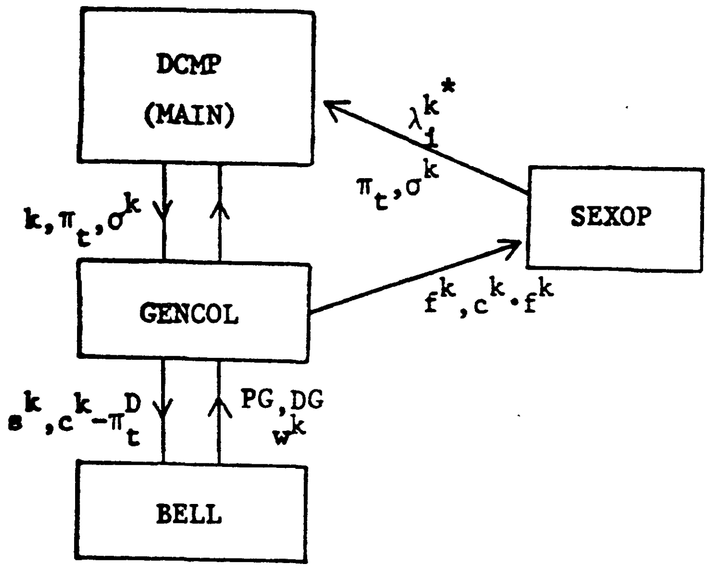


Figure 3

Interaction between RHSD and its subprograms

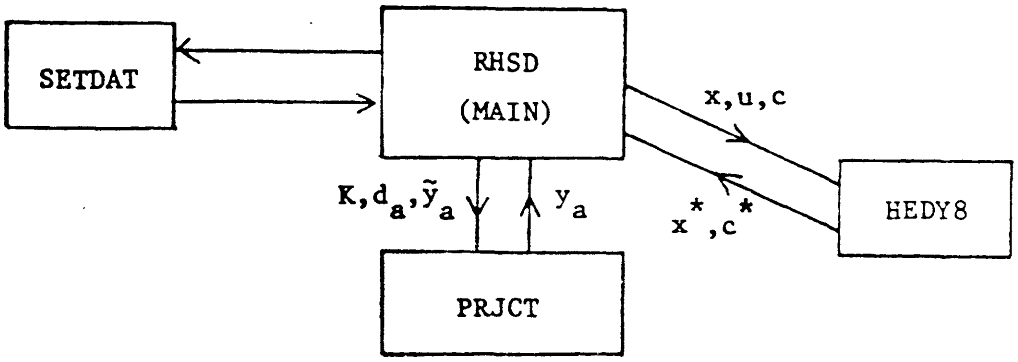


Table 1

Test Problems

Problem Name	Number of Arcs (NARC)	Number of Capacitated Arcs (NCAP)	Number of Nodes (NN)	Average Capacity per capacitated arc (AVCAP)
P1	98	98	47	6.0
P2	96	96	25	13.0
P3	204	95	83	12.0
P4	268	98	44	16.0

Table 2

Results for P1

Run Name	No. of Commodities (NCOM)	Total Flow (FTOT)	Optimal Value (VOPT)	Iterations to Optimality (ITOPT)	Time (sec.)
P1.1	3	28.0	486.0	7	.27
P1.2	5	"	478.0	5	.39
P1.3	7	"	492.0	6	.60
P1.4	6	30.0	518.0	7	.53
P1.5	10	"	498.0	5	.85
P1.6	15	"	497.0	5	.90
P1.7	3	28.0	482.0	12	.63
P1.8	3	28.0	470.0	8	.39

Table 3

Results for P2

Run Name	No. of Commodities (NCOM)	Total Flow (FTOT)	Optimal Value (VOPT)	Iterations to Optimality (ITOPT)	Time (sec.)
P2.1	4	120.0	4836.0	6	.50
P2.2	10	130.0	5364.0	7	1.73
P2.3	20	150.0	5852.0	6	4.31
P2.4	34	150.0	5840.0	4	3.13
P2.5	10	60.0	2304.0	1	0.04
P2.6	14	84.0	3192.0	1	0.06
P2.7	20	120.0	4522.0	3	0.57
P2.8	24	144.0	5602.0	4	1.37
P2.9	30	180.0	Infeasible	5	4.13

Table 4

Results for P3

Run Name	No. of Commodities (NCOM)	Total Flow (FTOT)	Optimal Value (VOPT)	Iterations to Optimality (ITOPT)	Time (sec.)
P3.1	4	105.0	2271.0	10	1.51
P3.2	6	"	2269.0	10	1.41
P3.3	12	"	2213.0	9	3.48
P3.4	18	"	2088.0	6	3.11
P3.5	6	36.0	712.0	2	0.12
P3.6	12	72.0	1444.0	4	0.81
P3.7	18	108.0	2155.0	9	2.80

Table 5

Results for P4

Run Name	No. of Commodities (NCOM)	Total Flow (FTOT)	Optimal Value (VOPT)	Iterations to Optimality (ITOPT)	Time (sec.)
P4.1	6	90.0	3640.0	2	0.20
P4.2	10	150.0	6006.0	4	0.58
P4.3	14	210.0	8858.0	5	1.44
P4.4	20	300.0	12488.0	3	1.70
P4.5	30	450.0	18162.0	4	3.98

IV. IMPLEMENTATION AND COMPUTATIONAL RESULTS

B - THE RESOURCE-DIRECTIVE DECOMPOSITION ALGORITHM

This chapter is devoted to the subgradient optimization algorithm as implemented to solve the multicommodity flow problem. We will describe the code we wrote for this purpose and discuss computational results.

1. Program Structure

The resource-directive decomposition algorithm involves a master problem, the main component of which is a projection algorithm called PRJCT; and subproblems which are solved by an Out-of-Kilter code called HEDY8. The main program is called RHSD (for right-hand-side decomposition). We now describe each component of the code in great detail.

The main program reads in the problem data and sets up the data structure required by HEDY8 by calling a subroutine SETDAT. The original problem data is manipulated by adding artificial return arcs (t^k,s^k) for each commodity k. The flow value F^k for that commodity may thus be viewed as the total flow on this added arc. As a result the flow for each commodity becomes a circulation flow where the net flow at any node is zero. These arcs are given cost c_o^k and capacity F_o^k in accordance with the subproblem definition in (II. 12.1 - 12.5). SETDAT also transforms multiple arcs between a pair of nodes - i.e. (i,j) and (j,i) - by inserting an additional artificial node in one of the arcs, thereby adding a new arc to the network. Thus the total number of arcs in the new network structure is

$$NA^* = NA + NCOM + NMULT$$

where NMULT is the number of multiple arcs (corresponding to undirected arcs in our case) in the original network. This transformation is required by HEDY8. Several pointer arrays needed by HEDY8 are also constructed by SETDAT.

HEDY8 is an efficient Out-of-Kilter algorithm which uses recent ideas in data manipulation and is found to be computationally superior to a number of commercial codes. The algorithm, as well as the required data structure is discussed in detail in [1, 2]. As used by our code, HEDY8 requires the specification of three vectors x, u, and c, of length NA^*, which contain the flow, the capacity (upper bound of flow), and the reduced cost for every arc in the network. Naturally, as for all primal-dual methods, flow values need not be feasible (i.e., may have $f_a > u_a$ on some arc a). The algorithm then looks for an optimal solution by maintaining dual feasibility and complementary slackness and striving for primal feasibility. The subroutine HEDY8 returns the optimal flows and reduced costs x^*, c^*.

PRJCT is the name given to the subroutine performing the projection (II.35) by using the algorithm outlined in equations (II.35-38). As noted there, the projection requires an initial ordering of the components of the input vector $\tilde{y}_a$ for arc a. This is done by using the heapsort algorithm given in Fortran in [13]. To call PRJCT one need only specify the parameters K, d_a and the capacity allocation vector $\tilde{y}_a$ for any ar a. PRJCT returns the solution y_a.

The interaction between program components is shown in Fig. 8. Upon constructing the data structures and the augmented network structure to be used by all subproblems;HEDY8 is called to solve each subproblem with the initial values of zero flow, the capacity vector y_o, and the original arc costs. Thereupon at a general iteration i, the subproblems are solved with right-hand-side $(u=)y_i^k$ for k=1,...,K. The dual variables $\gamma_i^k(y_i^k)$ are obtained from the optimal reduced costs of the subproblem by the relation:

$$\gamma_{i,a}^k = \text{Max}\ \{0, -\bar{c}_a^k\} \qquad (1)$$

where $\bar{c}_a^k$ is the optimal reduced cost on arc a. The dual variables in (1) determine the set of arcs T_i to be flagged for projection [see (II.39)] for which the subroutine PRJCT is called.

Note that at any iteration, only the right-hand-side of subproblem k is changed and this change occurs in relatively few entries of y_i^k. Thus the solution x^* and reduced costs c^* to the subproblem at iteration i are saved to serve as the starting solution at iteration i+1. Thus 2K vectors of length NA are saved which may demand substantial storage for large K but are believed to speed up the solution of subproblems substantially. Clearly when solving subproblem k only one artificial commodity return arc (t^k,s^k) is given a nonzero upper bound F_o^k. All the other return arcs have a capacity of 0 for commodity k. This simple rule allows us to use a single data structure - that of the full network - horizontally across the subproblems.

One problem in using HEDY8 to solve the subproblems was a mismatch of data types. HEDY8 accepts only integer values as data and is written in terms of integer arithmetic. The capacity allocation resulting from the projection routine are obviously real variables, and as such should be rounded off before they could be passed to HEDY8 as the vector u.

To achieve this we simply multiplied the capacities by an appropriate power of ten (depending upon the degree of accuracy required) and placed the result in the <u>integer</u> array u. We note that as a result, the optimal dual variables $\gamma_{i,a}^{k}$ defined in (1) above will also be integer and differ slightly from the real dual variables for the problem. Originally we were concerned whether this would affect the convergence properties of the algorithm. To check this an earlier version of RHSD was coded to use SEXOP, in place of HEDY8, to solve the subproblems.

A small test problem with 10 arcs, 6 nodes, and 2 commodities was chosen to carry out the comparison. All arcs were capacitated although on 5 arcs capacities were large enough not to be binding. Below we list the optimal value of the master and the step size obtained at the end of each iteration from the two versions S and H of the algorithm using SEXOP and HEDY8 respectively. In rounding-off, only 4 significant digits were used.

iteration	$\underline{v}_S$	$\underline{t}_S$	$\underline{v}_H$	$\underline{t}_H$
1	753.20	0.0140	752.82	0.0141
2	873.40	0.0072	875.70	0.0070
5	917.55	0.0062	917.49	0.0061
10	941.06	5.8941	941.07	5.8931
11	942.00	3.6250	942.00	5.8000

Convergence is obtained at iteration 11 for both algorithms. The intermediate results are also in close agreement as seen above. We also compared the two versions on a larger test problem of 26 arcs, 9 nodes, and 4 commodities. Again the discrepancies were insignificant.

Given that rounding off is no problem, the SEXOP option is inferior since it currently offers no possibility of saving the previous subproblem solution in core for more efficient re-optimization. The best one can do is to use the dual simplex method on the solution for subproblem k as a starting solution for subproblem k+1. When using HEDY8 however, we save the solution of each subproblem to start off the Out-of-Kilter algorithm for the <u>same</u> subproblem in the next iteration.

We shall now comment on our computational experience with the subgradient algorithm. The algorithm we presented in Chap. II has a number of controllable factors which one may vary when searching for the best algorithmic performance. These factors may be listed as:

1 - Choice of an initial capacity allocation $y_o \varepsilon S$

2 - Choice of the artifical costs c_o^k in the subproblem objective function.

3 - Choice of the estimate $\hat{v}$ to be used in (II.32).

4 - Choice of the step-size factors $\{\lambda_i\}$ in (II.32).

Most of our computational experimenting focused on issues 3 and 4 of the above. We shall comment on 1 and 2 rather briefly.

1 - <u>The initial capacity allocation</u> $y_o = (y_o^1,\ldots,y_o^K)$ was set by the following simple formula:

$$y_o^k = \frac{F_o^k}{FTOT} \cdot d \tag{2}$$

where FTOT is defined to be

$$FTOT = \sum_{k=1}^{K} F_o^k \tag{3}$$

as before. This choice is not a very sophisticated one but is definitely easy to compute. A more complex alternative would involve solving each of the subproblems with the original arc capacities (i.e. $y^k = d$) thus obtaining the solution to the multicommodity problem with the mutual capacity constraints (II.9.3) relaxed in favor of single-commodity constraints $f^k \leq d$ for all k. This "interaction-free" flow pattern can then be used as a basis for determining y_o. This choice seems attractive particularly when the flow-carrying chains for different commodities are expected to be essentially diverse from each other, since one may then expect the optimal flow pattern to be close to the interaction free pattern discussed above.

2 - <u>The artificial costs</u> c_o^k reflect the extent to which we are encouraging the variable F^k to achieve its upper bound F_o^k. Essentially we chose c_o^k values consistent with the artificial costs in the Dantzig-Wolfe decomposition algorithm. That is, we chose the costs per unit flow c_o^k to be around 10-20 times the average routing cost per unit flow. We had only one occasion to test the sensitivity of our code to this choice: When running RHSD on Problem P1.5 with 10 commodities we noticed that many flow values F^k were consistently below their upper bound ($F^k < F_o^k$) with a value of 200 for c_o^k. To encourage greater values of F^k, c_o^k was increased to 2000, and as a result, 4 commodities attained the required flow value F_o^k. The ratio VBEST/v^* (where VBEST denotes the best value obtained for the master problem) also increased from .79 to .88. We also found that increasing c_o^k leads to a decrease in the values of t_i's. However the general behavior of the algorithm is not affected by the choice of c_o^k.

3 - The subgradient algorithm requires an <u>underestimate</u> $\hat{v}$ of the

optimal value v^* to be used in the step-size formula (II.32). Held, Wolfe, and Crowder, however, report using an <u>overestimate</u> $\hat{v} > v^*$ for lack of a readily available underestimate. In our case, too, we decided to use an underestimate. We recall from Chap. II that

$$v^* = \sum_{k=1}^{K} c_o^k \cdot F_o^k - v_1^* \qquad (4)$$

where v_1^* are the minimal routing costs defined in (II.9.1). Assuming that c_o^k's are chosen to be equal ($c_o^k = c_o\ \forall k$), we may write (4) as

$$v^* = c_o \cdot \text{FTOT} - v_1^* . \qquad (5)$$

We see that by (5), any underestimate $\hat{v}_1$ of $v_1^* (\hat{v}_1 < v_1^*)$ will define an overestimate $\hat{v} = c_o \cdot \text{FTOT} - \hat{v}_1$ of v^*. A very crude overestimate may be obtained immediately by choosing $\hat{v}_1 = 0$, so that

$$\hat{v} = c_o \cdot \text{FTOT} . \qquad (6)$$

A much better estimate may be obtained by solving the interaction-free version of (II.9.1-9.3) discussed above where (9.3) is replaced by $f^k \leq d$ for all k. Denoting the objective function thus defined by $\hat{v}_D$, we clearly have $\hat{v}_o < v_1^*$ and obtain an overestimate

$$\hat{v} = c_o \cdot \text{FTOT} - \hat{v}_o . \qquad (7)$$

We have experimented with both types of overestimates (6) and (7) and obtained significantly better results with the tigher overestimate in (7). In two otherwise identical runs on P1.1 the best value for the master (VBEST) increased from 5097.4 to 5110 in 80 iterations as a result of using (7) instead of (6) ($v^* = 5114$ for this run). The improved behavior of the algorithm is specially apparent when values close to v^* are obtained.

4 - A critical issue in using a step-size sequence defined by (II.32) is the choice of the λ_i's values. Held, Wolfe, and Crowder [18] state that the choice of step sizes is imperfectly understood. The strategy they used for the multicommodity max flow problem was to set $\lambda = 2$ for 2N iterations (where N depends on problem size) and then halving both the values of λ and the number of iterations until the latter reaches a threshold value N_o, whereupon λ is halved every N_o iterations.

In order to understand the behavior of the algoirthm for different choices of the $\{\lambda_i\}$ sequence, we made a number of runs for the problem Pl.1. Table 6 summarizes the results. We shall now comment on the results according to the general strategy used for defining the λ_i's:

Constant λ.

We initially tried runs with a constant λ and obtained rapid convergence for small test problems with 10 and 26 arcs (2 and 4 commodities respectively). However for the larger Pl.1 problem (of 98 arcs) runs 1 and 2 show that such a strategy does not result in values close enough to the optimum of 5114. An overestimate of 5600 was chosen according to (6), thus the values for VBEST are expected to improve if we use $\hat{v} = 5116$ following (7).

Successive Halving of λ.

Here we followed the suggestion of Held, Wolfe, and Crowder [18] discussed before. Runs 3 and 4 show marked improvement over runs 1 and 2 though they also use $\hat{v} = 5600$. Note that comparing runs 3 and 4 shows the iteration numbers chosen for halving λ will also affect the best objective function value obtained. The improvement of run 5 over run 4 is due only to employing the better $\hat{v}$ value of 5116.

<u>Adaptive Control of λ</u>.

The above strategy of halving λ depends upon the choice of a sequence of iteration numbers at which λ is halved. The choice of such a sequence (or N and N_o in our earlier discussion) requires some hindsight which is problem-dependent. We shall now consider a scheme which ensures decreasing the size of λ and does not require the prespecification of such a sequence.

To motivate our adaptive strategy, we present the iteration values and step sizes resulting from a constant $\lambda = 2$ value in Table 7. The last column represents the ratio of current t_i to the last step size obtained. We see from Table 7 that as soon as a good value of v_i is obtained for the master problem, the corresponding t_i becomes relatively large. This is probably due to the fact that most $\gamma_{i,a}^{k}$'s become small or vanish, thus making the denominator of (II.32) small. The net effect is that the next perturbation in the capacity vector y_i will be too large causing it to move away to a bad value resulting in a sudden drop in the objective function value v_i. This happens at iterations 29, 35, and 54 of Table 7. It is thus necessary that λ_i's get smaller with increasing i to offset such large fluctuations. This phenomenon in part explains the efficacy of the halving strategy discussed above. The adaptive strategy takes two measures to control the step sizes:

a) It decreases the current t_i when it is deemed to have too large a value.

b) It decreases the value of λ be used in the successive steps. The tracking signal for the adaptive strategy is the ratio t_i/t_{i-1} reported in Table 7 for Run 1. Thus we may summarize it as:

If the ratio $t_i/t_{i-1} \geq R$ (some threshold); reset

$$t_i \leftarrow a \cdot t_i \tag{8}$$

$$\lambda \leftarrow \alpha \cdot \lambda . \tag{9}$$

Some results for adaptive strategies are shown in runs 6-8. In all cases $\hat{v} = 5600$, and $a = 0.01$. α should not be too small since then two or three triggerings of the scaling operation (9) would result in an overly small value of λ resulting in a very small sequence t_i. This, in turn, would mean that each time the vector y_i is perturbed by an insignificant amount so that the v_i values will change very slowly with i. The choice of $\alpha = 0.01$ in run 6 resulted in this behavior.

To us, the adaptive strategy seems to be a computationally attractive alternative to the halving strategy. In all cases it resulted in a better sequence of values v_i which came closer to the optimum v^*.

2. Timing the Projection Routine

As in the case of the Dantzig-Wolfe decomposition code, we found it useful to time the subroutine which solves the projection problem to identify whether much time is spent performing the projections.

Recall that the input parameters for PRJCT were NCOM (or K) and d_a, the sum of the components of the projection vector y_a. In timing PRJCT we varied NCOM and set d_a = NCOM. Thus we project K-vectors on the hyperplane defined by

$$\{x \in R^K \mid \sum_{k=1}^{K} x^k = K \mid x^k \geq 0 \ \forall k\} .$$

The results are shown below

NCOM	3	5	10	20	50
Average time (in 10^{-4} sec.) per projection	1.13	1.92	4.08	9.34	27.08

As remarked earlier not all arcs are projected. Let NP_i be the number of arcs projected at iteration i. We observed that NP_i/NARC decreases with i once we enter a reasonable neighborhood of the optimum. Again this is due to many $\gamma^k_{i,a}$'s vanishing once the optimum is neared. Also one would expect NP_i to depend on NCOM: The more commodities there are, the greater the probability of $\gamma^k_{i,a}$ being nonzero for some k.

In a test problem with 26 capacitated arcs and 4 commodities we computed an average of $\overline{NP}_i = 15$ for $1 \leq i \leq 10$, and 8 for $10 \leq i \leq 19$. The algorithm converged in 19 iterations. Assuming the ratio NP_i/NARC to be 0.5, at worst, for the 3 commodity problem P1.1; we would have to spend approximately 0.7 seconds for projections in the course of 120 iterations. We have timed RHSD on P1.1 to take approximately 6 seconds. So approximately 12% of the solution time goes into projections.

3. General Behavior of the Algorithm

We have described the effect of various strategies on the convergence exhibited by the right-hand-side allocation algorithm (RHSD). We noted that with a good overestimate $\hat{v}$ and an adaptive strategy for the step size one may get close approximations to the optimal value v^*. It may be argued, however, that the output of interest in the optimal flow pattern f^{k*} for each commodity k, rather than merely the optimal value. Moreover if the capacities are integral we would like to obtain integral values for the optimal flow of each commodity on each arc. The Dantzig-Wolfe decomposition code exhibits this property regularly. Unfortunately resource-directive decomposition will generally fail to do this since the arc capacities allocated to each commodity result from a projection and may rarely be expected to be integral. To give only one example, the optimal flow pattern obtained by Dantzig-Wolfe decomposition on P1.1 for commodity 1 loads three chains C_1, C_2, and C_3 with chain flows 5, 3, and 1 respectively. The load pattern obtained from the best solution to P1.1 with RHSD places the values 4.4618, 2.9937, and 1.5445 on C_1, C_2, and C_3; and in addition chooses a fourth chain C_4 with a flow value of .0063. Thus RHSD does not yield integral solutions though it essentially chooses the correct chains. Here there may be some hope of reallocating the flows, once the chains are identified, to achieve integrality if needed.

The behavior of RHSD worsens as the number of commodities increases. For 10 commodities the objective function value is far from the optimum and half the commodities fail to meet the flow requirements. Thus RHSD should be limited to networks with a small number of commodities although largest network sizes are not expected to affect the convergence of RHSD much. The solution times are of the order of 6 sec. for 120 iterations on a 3 commodity version of P1.

Table 6

Step Size Strategies for RHSD

Run Name	Strategy	Number of Iterations	Best Value Obtained	Iteration Number of Best Value.
1	$\lambda = 2$ constant	120	5084.9	54
2	$\lambda = 1$ constant	60	5080.6	52
3	λ halved at 60 & 90	120	5088.2	70
4	λ halved at 50, 75	80	5097.4	73
5	λ halved at 50, 75, 90 (used better $\hat{v}$)	120	5112.4	120
6	Adaptive with $\alpha = 0.01$	120	5103.7	120
7	Adaptive with $\alpha = 0.3$	120	5112.9	95
8	Adaptive with $\alpha = 0.5$	120	5111.8	120

Table 7

Results of Run 1 of RHSD

Iteration	v_i	t_i	t_i/t_{i-1}
1	2494.2	0.039	-
10	4312.8	0.027	1.70
20	4813.4	0.017	1.74
28	4934.1	0.005	0.35
29	5080.7	1.927	368.09
30	2421.4	0.017	0.01
34	4927.0	0.009	0.53
35	5076.9	1.718	191.0
36	2661.2	0.011	0.006
53	4743.9	0.027	.64
54	5084.9	7.518	278.45
55	1440.2	0.018	0.002

V. THE MULTIFLEET ROUTING PROBLEM : AN APPLICATION

In this chapter we describe an application of the decomposition algorithm of Chapter II to solve a Multifleet Routing Problem.

1. The Routing Problem

We start with a brief description of the problem, following [25-27]. The problem may be presented as follows : Given a schedule map of potential non-stop services, what set of services should be flown to maximize system income and how many aircrafts are required. The system income is the sum of revenues associated with the services minus ownership costs attached to each aircraft.

We now elaborate on these elements : A schedule map is a space-time network associated with the system specifying all possible schedules for an aircraft. As an example consider Fig 4a which exhibits the spatial route map between three cities, A, B, and C. Any flight from, say, A to B departs from A at a given time and arrives at B at a later time. Thus we may associate with each city a finite set of arrival and departure times shown as nodes on the vertical segments A, B and C in Fig 4b. A possible flight from, say, A to B is shown by a horizontal arc between the departure and arrival nodes. Thus we have a potential service leaving city A at 12:00 hours and arriving at B at 18:40 hours. Such horizontal arcs are called flight or service arcs. The vertical arcs may be called ground arcs. They allow the aircraft to remain on the ground awaiting a departure (Time flows vertically downwards in our diagram). Finally the third class of arcs are cycle arcs (or overnight arcs). These allow the fleet to return to the next schedule cycle. This is justified by our assumption of system

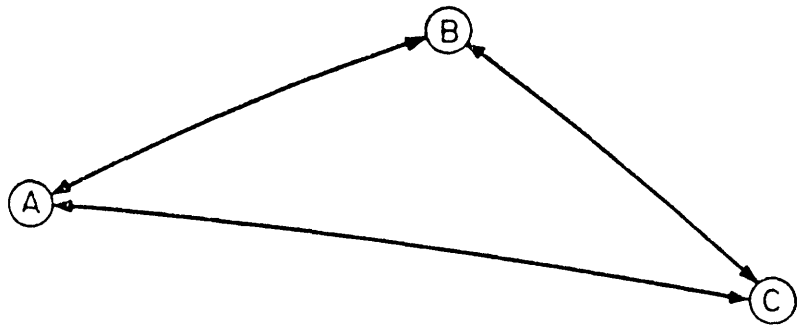


FIG.4a ROUTE MAP

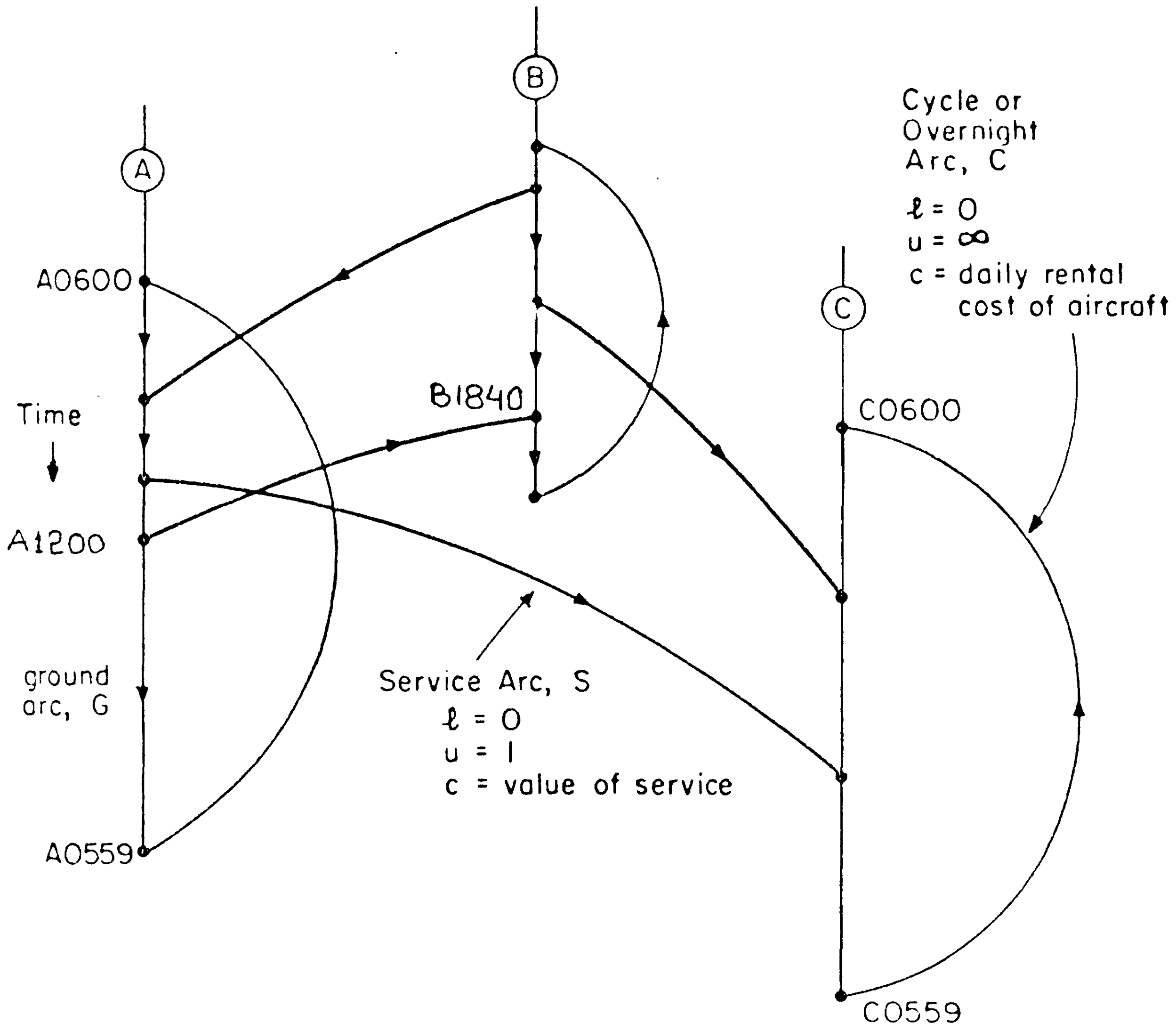


FIG.4b SCHEDULE MAP

periodicity : At the end of each cycle all planes are available for the next cycle.

For a single aircraft type the problem described above may be solved by the Out-of-Kilter algorithm. In practice, however, there are vehicles of varying size and operating costs. Since the flow of each aircraft must be distinguished from other types, consideration of several aircraft types involves a multicopy network model which we now formulate as a m.c. flow problem.

2. Problem Formulation

We take the schedule map for our network $G = (N,A)$. Each node $i \in N$ thus corresponds to a city and a specified time. The arc set A is the disjoint union of three sets A_s, A_g, A_c denoting the set of service, ground, and cycle arcs respectively.

Let f_a^k = number of aircrafts of type k on arc a.

and c_a^k = cost per unit flow of aircraft type k on arc a.

Then the problem is

$$\text{Max} \quad z = \sum_{k=1}^{K} c^k \cdot f^k \qquad (1.1)$$

$$\text{s.t.} \qquad \tilde{E}^k \cdot f^k = 0 \qquad \forall k \qquad (1.2)$$

$$\ell^k \leq f^k \leq u^k \qquad \forall k \qquad (1.3)$$

$$\sum_{k=1}^{K} f_a^k \leq 1 \qquad \forall a \in A_s \qquad (1.4)$$

We have considered each aircraft type as a commodity indexed by k.

Constraint (1.2) is simply flow conservation for circulation flow of each commodity. (1.3) imposes upper and lower bounds for flow of a single type on each arc. (1.4) is the complicating capacity or bundle constraint. Together with integrality of f_a^k it ensures that each service (flight) is flown, if at all, by only one aircraft. To have physically meaningful results the flows should be integral so that (1.1-4) should be viewed as the linear programming relaxation of an integer program. If the optimal solution to this relaxation is also integer then the original problem is solved. Previous computational results [25], [26] for this type of problems indicate that integral solutions often occur.

To put (1.1-4) in a form compatible with our code we proceed as follows : We remove all cycles (or overnight arcs) and instead attach a supersource and a supersink where all aircrafts originate and terminate respectively. These act as the common OD pair (s,t) for all commodities k. Thus the topmost node of each vertical line (city) in the schedule map is joined to the supersource s and the lowest node to t. The resulting network is then acyclic. The ownership costs c_a^k attached to cycle arcs $a \in A_c$ may now be attached to the arcs emanating from s. The arcs incident to t are given a cost of zero. The flow value F^k denotes the total number of vessels of type k utilized. The problem may then be stated as

$$\text{Min} \quad z = \sum_{k=1}^{K} c^k \cdot f^k \tag{2.1}$$

$$\text{s.t.} \qquad E f^k - g^k F^k = 0 \qquad \forall k \tag{2.2}$$

$$f^k, F^k \geq 0 \qquad \forall k \tag{2.3}$$

$$\sum_{k=1}^{K} f_a^k \leq 1 \qquad \forall a \in A_s \tag{2.4}$$

In (2.1) we minimizes losses. When expanded the objective function has the form

$$z = \sum_{k=1}^{K} (c_o^k F^k + \sum_{a \varepsilon A_g} c_a^k \cdot f_a^k - \sum_{a \varepsilon A_s} c_a^k \cdot f_a^k) \qquad (3)$$

where c_o^k = ownership cost attached to an aircraft of type k. Thus the first term inside the parentheses in (3) refers to ownership costs, the second reflects ground waiting costs, while the third denotes revenues from flights. Notice that we have chosen the lower bound $\ell^k = 0$ and have imposed no upper bounds. We also note that the set

$$P^k \triangleq \{(f^k\ F^k) \mid E\, f^k = g^k F^k,\ \ f^k, F^k \geq 0\} \qquad (4)$$

is a cone. Any feasible flow (satisfying 2.2-3) may be expressed as

$$f^k = \sum_{i=1}^{I^k} \rho_i^k\, f_i^k$$

with $(f_i^k, 1)$ an extreme ray of P^k. Thus our master problem may be stated as

$$\text{Min} \quad \sum_{k=1}^{K} \sum_{i=1}^{I^k} \hat{c}_i^k \cdot \rho_i^k \qquad (5.1)$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \sum_{i=1}^{I^k} \rho_i^k\, f_{i,a}^k \leq 1 \qquad a \varepsilon A_s \qquad (5.2)$$

$$\rho_i^k \geq 0 \qquad \forall i, \forall k \qquad (5.3)$$

$$\text{where} \quad \hat{c}_i^k = c^k \cdot f_i^k = c_o^k + \sum_{a \varepsilon A_g} c_a^k \cdot f_{i,a}^k - \sum_{a \varepsilon A_s} c_a^k \cdot f_{i,a}^k \qquad (6)$$

Note that we have no convexity constraints (as in II.4.3) since we are decomposing over a cone.

As before attach dual variables π to (5.2). The criterion for entering a column into the restricted master is

$$\hat{c}_i^k - \pi D f_i^k < 0$$

where D is a matrix transforming f_i^k to a subvector of flows on A_s only. Thus the subproblems are, as before, shortest path problems with respect to the arc costs $(c^k - \pi D)$. There are however two minor changes : a) the arc costs are not all nonnegative and some arcs have negative lengths, b) the network structure is acyclic. We know that condition (a) causes no problems as long as no negative cycles exist, which condition (b) insures so that the BELL algorithm may be applied with impunity. Further the shortest path algorithm may be accelerated by taking the acyclic structure into account. We have not adopted this course due to the small percentage of time our code expends on the subproblems as opposed to the master problem.

3. Implementation and Computational Results

The decomposition code of Chapter III was modified slightly to incorporate the changes given above. We shall refer to its new version as DCMP2. Note that in contrast to old version DCMP, the flow values F^k are now decision variables.

The problem solved by DCMP2 has the route map shown in Fig. 5 and thus involves 5 cities A - E.

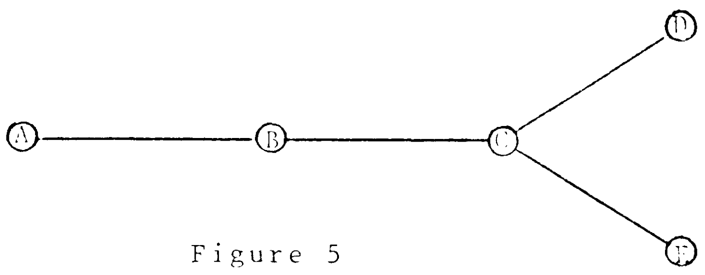


Figure 5

Figure 6 shows the complete schedule map (as taken from [27]). There are 90 nodes, 58 service arcs, 85 ground arcs, and 5 cycle arcs. Upon adding supernodes and replacing the cycle arcs for each city by arcs emanating from the supersource and terminating at the supersink, we will have two more nodes and ten arcs. Obviously the number of ground and service arcs remain unchanged. Thus in our notation the problem parameters are :

$$NN = 92, \quad NA = 58 + 85 + 10 = 153, \quad NCOM = 2$$

Only two aircraft types are used $(K = NCOM = 2)$ and the only capacitated arcs are the service arcs $NCAP = |A_s| = 58$. Reference [27] gives a full description of the problem (called the Tech. Airways Problem).

We shall now describe the solution obtained by DCMP2 : We start by noting that each column $(f_i^k, 1)$ describes a chain from s to t with unit flow which fully describes the schedule of an aircraft of type k during the planning cycle. Thus each vessel starts from a certain city at the beginning of our time cycle and, upon traversing a combination of service and ground arcs (corresponding to flight and waiting periods), terminates its activity in some city at the end of the time cycle. To describe the optimal solution we need only list those basic columns at optimality with strictly positive flow. Note further that for integrality we need each f_i^k to be integer otherwise fractional flows will enter the picture.

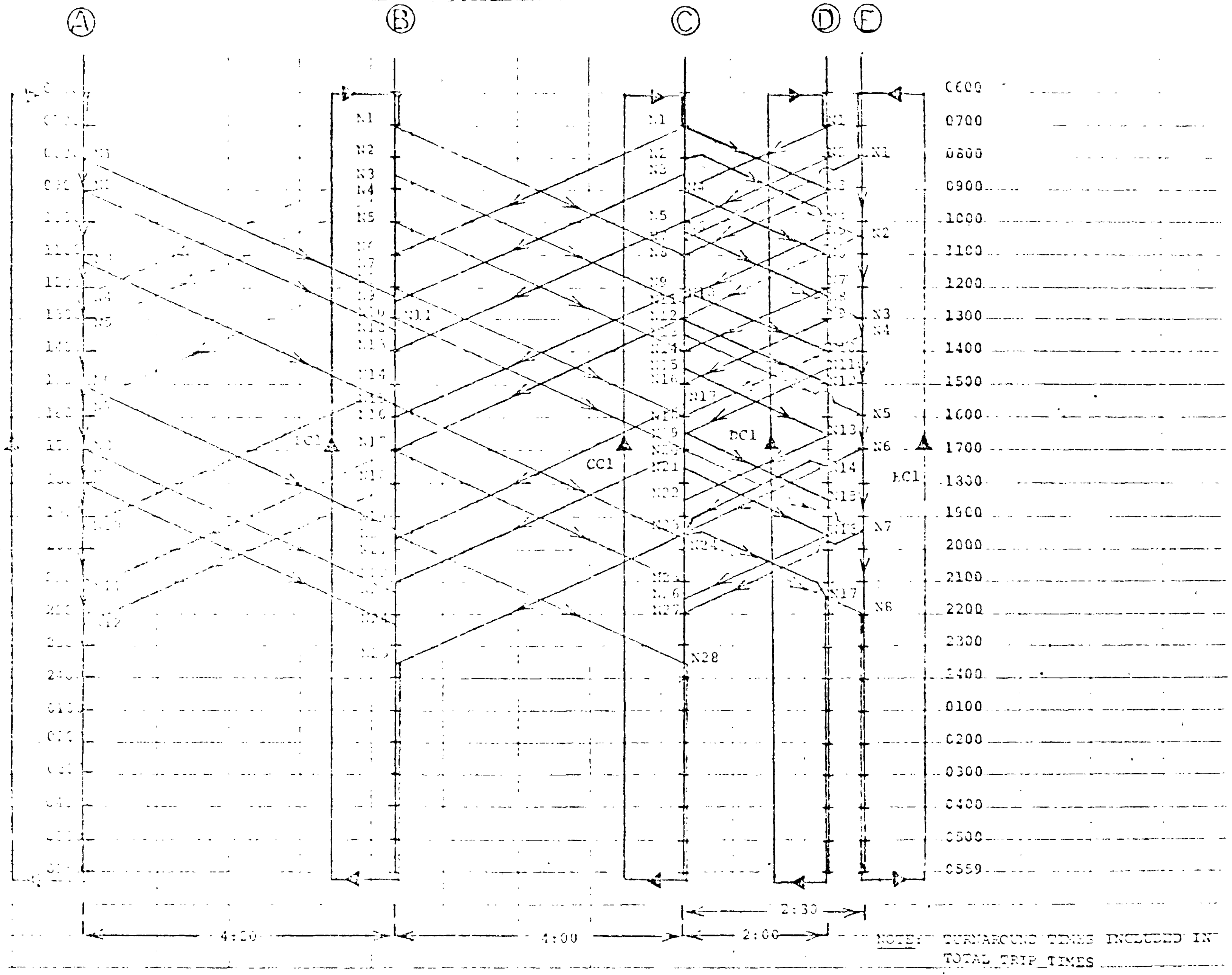
TECH AIRWAYS- SCHEDULE MAP NETWORK

[illegible]

90 Nodes
5 Cycle Arcs
[illegible]
85 [illegible]
5 Cities
8 City-pair Markets

Total Arcs-145

[illegible]



NOTE: TURNAROUND TIMES INCLUDED IN TOTAL TRIP TIMES

Figure 6

[REDUCED COPY]

The following schedules combine to form the optimal solution :

(S1) C1 - D3 - C8 - C13 - E5 - E6 - C23 - D17

(S2) D1 - D2 - C5 - B13 - B17 - A11 - A12

(S3) B1 - B2 - A4 - A8 - B23 - B25

(S4) D1 - C4 - D6 - D14 - C23 - E8

(S5) B1 - B5 - C14 - C21 - D16 - C26 - C28

(S6) C1 - C2 - E2 - C12 - B17 - C25 - C28

(S7) E1 - C7 - C9 - B16 - B18 - A12

(S8) C1 - B6 - B7 - A7 - A9 - B24 - B25

(S9) A1 - B9 - B11 - C20 - E7 - C27 - C28

To explain the notation we examine (S4) more carefully : The aircraft starts in city D at node D1 , then flies to C4 followed by another flight back to D6. Then there is a waiting period in city D characterized by D6 - D14 (all intermediate nodes are deleted for simplicity), followed by two flights first to C23 , then to E8. Thus the schedule of the aircraft is completely specified during the planning horizon of one cycle.

The aircraft type is also specified for each schedule (the index k gives the type) and in our solution schedule (S2) corresponds to a type 2 aircraft). All other aircrafts (8 in number) are of type one. The total fleet size is $F^1 + F^2 = 8 + 1 = 9$.

The optimal value for the objective function was found to be z = \$ 17,000 corresponding to zero costs on ground arcs and ownership costs of \$ 1,500 and \$ 2,000 for aircraft types 1 and 2 respectively. Altogether 31 services are flown.

Table 8 shows the convergence of DCMP2 for this problem. The optimality criterion is satisfied after 22 iterations of the master problem. The CPU time excluding I/O on the IBM 370/168 machine was 0.67 seconds which is most encouraging for this type of problem. Another encouraging (though not unexpected) result was the integrality of the optimal solution.

Table 8

Convergence Behavior for the Multifleet Problem

Iteration	Objective Function
1	6,400
2	6,400
3	6,400
4	8,250
5	8,250
6	10,050
7	11,550
8	11,550
9	11,550
10	11,650
11	11,650
12	12,950
13	12,950
14	13,550
15	14,650
16	15,650
17	15,850
18	16,000
19	16,000
20	17,000
21	17,000
22	17,000 (Terminates)

VI. CONCLUSIONS

We may classify exact algorithms for the m.c. flow problem into three classes : price-directive decomposition, resource-directive decomposition, and compact inverse methods (or primal codes). In our computational experiments we have tested two codes, one price-directive and one resource-directive. Neither requires prohibitive implementation efforts and both may be built around readily available Linear Programming and Out-of Kilter codes.

We found that Dantzig-Wolfe decomposition works very well. Our results combined with some previous experience [24] indicate that price-directive methods clearly outperform resource-directive methods for these applications. However the subgradient resource-directive algorithm may still be recommended for very large networks (with few commodities) which Dantzig-Wolfe decomposition cannot handle due to the size of the master problem.

Further research in this area could concentrate on efficient implementation of primal codes especially in view of the recent advances in utilizing the problem structure and list-processing techniques as described in [20]. Using BOXSTEP [15,16] to solve m.c. problems may also be investigated profitably.

The multifleet routing problem described in Chapter V is typical of a large number of potential applications of m.c. models. The encouraging results hold promise for solving real scheduling problems in a variety of contexts described in [25].

Postscript

Upon completion of the computational experience on which this report is based [23], the author came across two papers by Kennington who has solved multicommodity capacitated transportation problems by two algorithms : a primal GUB approach [28] and a subgradient procedure similar to ours [29]. The networks he experiments with are somewhat smaller but the results for the subgradient algorithm generally resemble our conclusions.

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to Professor T.L. Magnanti for his supervision, careful readings, and numerous helpful comments. I thank Professor R.E. Marsten for the help he gave me with SEXOP. My friends Bruce L. Golden and Hedayat Z. Aashtiani were especially helpful with their shortest path and Out-of-Kilter codes. I was introduced to the Multifleet Routing Froblem by Professor R.W. Simpson and thank Peter Kivestu for providing me with the data and additional information. Finally I acknowledge the partial support of the U.S. Department of Transportation under the contract DOT-TSC-1058.

ADDITIONAL REFERENCES

[1] Aashtiani, H. Z.; "Solving Large Scale Network Optimization Problems by the Out-of-Kilter Method", M. S. Thesis, Sloan School of Management, M.I.T. (Feb. 1976).

[2] ______, and T. L. Magnanti; "Solving Large Scale Network Optimization Problems", presented at ORSA/TIMS National Meeting, Chicago, May 1975.

[3] Assad, A. A.; "Multi-commodity Network Flows - A Survey", Oper. Res. Center, M.I.T., Working Paper OR 045-75 (Dec. 75).

[4] Held, M., and R. M. Karp; "The Traveling-Salesman Problem, Part II", Math. Prog., 1, 1, 6-25 (Oct. 1971).

[5] Oettli, W.; "An Iterative Method, Having Linear Rates of Convergence, for Solving a Pair of Dual Linear Programs", Math. Prog. 3, 302-311 (1972).

[6] Barr, R. S., F. Glover, and D. Klingman; "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes", Math. Prog., 7, 60-86 (1974).

[7] Glover, F., D. Karney, D. Klingman, and A. Napier; "A Computations Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems", Man. Sci. 20, 5, 793-813 (1974).

[8] ______, D. Klingman, and J. Stutz; "An Augmented Threaded Index Method for Network Optimization", INFOR, 12, 293-813 (1974).

[9] Mulvey, J.; "Developing and Testing a Truly Large Scale Network Optimization Code", presented at ORSA/TIMS National Meeting, Chicago, May 1975.

[10] Golden, B.; "Shortest Path Algorithms: A Comparison", to appear in Oper. Res.

[11] Gilsinn, J. and C. Witzgall; "A Performance Comparison of Labelling Algorithms for Calculating Shortest Path Tress", National Bureau of Standards, Technical Note 772 (May 1973).

[12] Florian, M., et al.; "The Engine Scheduling Problem in a Railway Network", Université de Montréal, Département d'Informatique, Publication #133, (Oct. 1972) revised Nov. 1974.

[13] Nijenhuis, A., and H. S. Wilf; Combinatorial Algorithms, Academic Press, New York, pp. 225-31 (1975).

[14] Marsten, R. E.; "Users Manual for SEXOP", Release 4, Sloan School of Management, M.I.T. (February 1974).

[15] ______, "The Use of BOXSTEP Method in Discrete Optimization", Math. Prog. Study 3 (1975).

[16] ______, W.W.Hogan, and J.W.Blankenship; "The BOXSTEP Method for Large-Scale Optimization", Oper.Res. 23, 389-405 (1975).

[17] Tomlin, J.A.; "Minimum Cost Multicommodity Network Flows", Oper. Res., 14, 45-51 (1966).

[18] Held, M., P. Wolfe, and H.P. Crowder; "Validation of Subgradient Optimization", Math. Prog., 6, 1, 62-88 (Feb. 1974).

[19] Lasdon, L.S.; Optimization Theory for Large Systems, Macmillan, New York (1970).

[20] Magnanti, T.L.; "Optimization for Sparse Systems", pp.147-176 in Sparse Matrix Computations (J.R. Bunch & D.J. Rose, eds.), Academic Press, New York (1976).

[21] Geoffrion, A.M.; "Primal Resource Directive Approaches for Optimizing Nonlinear Decomposable Systems", Oper. Res., 18, 375-403 (1970).

[22] Grinold, R.C.; "Steepest Ascent for Large Scale Linear Programs", SIAM Review, 14, 3, 447-463 (July 1972).

[23] Assad, A.A.; "Solution Techniques for the Multicommodity Flow Problem", unpublished M.S. Thesis, Oper. Res. Center, MIT (June 1976).

[24] Swoveland, C.; "Decomposition Algorithms for the Multicommodity Distribution Problem", Western Man. Sci. Ins., Univ. of Calif. (LA), Working Paper #184 (Oct. 1971).

[25] Simpson, R.W.; "Scheduling and Routing Models for Airline Systems"; Flight Transportation Lab Report FTL-R68-3, MIT (Dec. 1969).

[26] Levin, A.; "Some Fleet Routing and Scheduling Problems for Air Transportation Systems", Flight Transportation Lab Report FTL-R68-5, MIT (Jan. 1969).

[27] Cohen, R.; Term paper for the course : Flight Transportation Oper. Analysis (May 1972).

[28] Kennington, J.L.; "Solving Multicommodity Transportation Problems Using a Primal Partitioning Simplex Technique", Dept. of Ind. Eng. & Oper. Res., Southern Methodist Univ. Tech. Report CP 75013 (revised May 1976).

[29] _______ , and M. Shalaby; "An Effective Subgradient Procedure for Minimal Cost Multicommodity Flow Problems", ibid., Tech. Report IEOR 750010 (revised March 1976).