European Journal of Operational Research 36 (1988) 107-115
North-Holland

Theory and Methodology

The equal flow problem

Agha Iqbal ALI
Department of General Business, University of Massachusetts at Amherst, Amherst, MA 01003, USA

Jeff KENNINGTON
Southern Methodist University, USA

Bala SHETTY
Texas A & M University, USA

Abstract: This paper presents a new algorithm for the solution of a network problem with equal flow side constraints. The solution technique is motivated by the desire to exploit the special structure of the side constraints and to maintain as much of the characteristics of pure network problems as possible. The proposed algorithm makes use of Lagrangean relaxation to obtain a lower bound and decomposition by right-hand-side allocation to obtain upper bounds. The Lagrangean dual serves not only to provide a lower bound used to assist in termination criteria for the upper bound, but also allows an initial allocation of equal flows for the upper bound. The algorithm has been tested on problems with up to 1500 nodes and 6000 arcs. Computational experience indicates that solutions whose objective function value is well within 1% of the optimum can be obtained in 1%–65% of the MPSX time depending on the amount of imbalance inherent in the problem. Incumbent integer solutions which are within 99.99% feasible and well within 1% of the proven lower bound are obtained in a straightforward manner requiring, on the average, 30% of the MPSX time required to obtain a linear optimum.

Keywords: Networks, algorithms, computational mathematical programming, decomposition, Lagrangean relaxation

1. Introduction

This paper makes use of relaxation in conjunction with decomposition for the solution of the equal flow problem. The problem is easily conceptualized as a minimal cost network flow problem with additional constraints on certain pairs of arcs. Specifically, given pairs of arcs are required to take on the same value. The problem is defined on a network presented by an $m \times n$ node-arc incidence matrix, A, in which K pairs of arcs are identified and required to have equal flow. Mathematically, this is expressed as:

$$\begin{array}{ll} \text{Min} & \boldsymbol{cx}, \\ \text{s.t.} & \boldsymbol{Ax} = \boldsymbol{b}, \\ & x_k = x_{k+K}, \quad k = 1, 2, \dots, K, \\ & \boldsymbol{0} \leqslant \boldsymbol{x} \leqslant \boldsymbol{u}, \\ & \boldsymbol{x} \text{ integer}, \end{array}$$

This research was supported in part by the Air Force Office of Scientific Research under Contract Number AFOSR 83-0278, the Department of Defense under Contract Number MDA 903-86-C-0-182, and Rome Air Development Center under Contract Number SCEEE PDP/86-75.

Received December 1986; revised May 1987

where c is a $1 \times n$ vector of unit costs, b is an $m \times 1$ vector of node requirements, $\mathbf{0}$ is an $n \times 1$ vector of zeroes, x is an $n \times 1$ vector of decision variables, and u is an $n \times 1$ vector of upper bounds. This mathematical statement of the problem, henceforth referred to as problem I1, assumes that the first $2K$ arcs appear in the equal flow constraints. This is not a restrictive assumption since by rearranging the order of the arcs, any equal flow problem with K pairs can be expressed in the above form. Note that the K pairs of arcs are mutually exclusive, i.e., an arc appears in at most one side constraint. We also assume without loss of generality, that $u_k = u_{k+K}$ for $k = 1, 2, \ldots, K$.

Applications of the equal flow problem include crew scheduling [5], estimating driver costs for transit operations [14], and the two duty period scheduling problem [11]. When integrality constraints are not present, the model is referred to as the linear equal flow problem (P1). The linear model is applicable to problems where integrality is not restrictive. For example, in federal matching of funds allocated to various projects [4]. The linear equal flow problem may be solved using a specialization of the simplex method for networks with side constraints [3]. It has also been solved by transformation to a nonlinear programming problem [4].

The use of relaxation techniques and/or decomposition techniques in the solution of problems with special structure in the constraint set is motivated by potential computational efficiencies. Glover, Glover and Martinson [6] address a generalized network problem in which arcs in specified subsets must have proportional flow. The solution approach is via solution of a series of problem relaxations and progressive bound adjustment. The underlying principle is shared in the ensuing development for the equal flow problem.

Lagrangean relaxation has been used to aid in the solution of the integer equal flow problem in two specific instances. Shepardson and Marsten [11] reformulate the two duty period scheduling problem as a single duty period scheduling problem with equal flow side constraints and integrality constraints on the variables. Turnquist and Malandraki [14] model the problem of estimating driver costs for transit operations as an integer equal flow problem. In both studies, the side constraints are dualized and the Lagrangean dual solved using subgradient optimization to yield a lower bound on the optimal objective value. In [14] step-size determination during the subgradient optimization process is aided by a line search.

The objective of this investigation is to develop and computationally test a new algorithm, based on relaxation and decomposition, for the linear equal flow problem and its use in solving the integer equal flow problem. The linear equal flow problem is a natural relaxation for the integer problem and also provides an approximation to the integer model. Because the problems are very closely allied, primarily due to the unimodularity of the node-arc incidence matrix, solutions to the integer model can be obtained by using a slight modification of the technique for the linear model. By employing relaxation and decomposition, solution of the equal flow problem is via two sequences of pure network problems, totally eliminating the computational overhead associated with maintaining the inverse of a basis matrix. The exploitation of the special structure of the side constraints and the network structure results in a decrease in both computer storage and computation time since reoptimization procedures are applicable for solution of subproblems of the two sequences.

The solution technique consists of making use of the Lagrangean dual of the equal flow problem with the side constraints dualized to obtain a lower bound. The Lagrangean relaxation of the equal flow problem does not enforce the equal flow constraints. The Lagrangean dual for the linear and the integer equal flow problem is exactly the same, since the constraint set for the Lagrangean relaxation is identical. This Lagrangean dual is similar to the quadratic programming problem used in [4]. The similarity lies in penalizing the violating equal flow constraints.

Upper bounds are obtained by use of a decomposition of the equal flow problem based on parametric changes in the requirements vector. The Lagrangean dual provides a lower bound which is used to aid the solution of the decomposition model in determining an initial right-hand-side allocation as well as providing a lower bound on the objective so that a solution is known to be within a percentage of the optimal. As such, the algorithm can terminate when a solution with a prespecified tolerance on the objective function value is obtained. By enforcing that the parametric changes in the requirements vector be such that integral allocations of equal flow be obtained, upper bounds on the integer problem can be obtained.

The solution technique makes use of subgradient optimization in the solution of both the Lagrangean dual for obtaining a lower bound and the decomposition model for obtaining the upper bound. Both the lower and upper bounding algorithms have been developed in the context of the general subgradient algorithm which is briefly presented in Section 2. Section 3 introduces the Lagrangian dual for the equal flow problem and the lower bounding algorithm. Section 4 presents the decomposition of the linear equal flow problem and the upper bounding algorithm. The overall procedure which makes use of the algorithms of Sections 3 and 4 is given in Section 5, computational results are given in Section 6 and conclusions drawn in Section 7.

2. The subgradient algorithm

The subgradient algorithm was first introduced by Shor [13] and provides a framework for solving nonlinear programming problems. It may be viewed as a generalization of the steepest descent (ascent) method for convex (concave) problems in which the gradient may not exist at all points. At points at which the gradient does not exist, the direction of movement is given by a subgradient. Subgradients do not necessarily provide improving directions and consequently, the convergence results of Zangwill [15] do not apply. Convergence of the subgradient algorithm is assured, however, under fairly minor conditions on the step size.

Given the nonlinear program P0,

$$\begin{array}{ll}\text{Min} & f(y),\\ \text{s.t.} & y \in G,\end{array}$$

where f is a real-valued function that is convex over the compact, convex, and nonempty set G. A vector η is a *subgradient* of f at y' if $f(y) - f(y') \geqslant \eta(y - y')$ for all $y \in G$. For any given $y' \in G$, the set of all subgradients of f at y' is denoted by $\delta f(y')$. Moving a sufficiently large distance s along η can yield a point $x = y' - s\eta$ such that $x \notin G$. The projection of the point x onto G, denoted by $P[x]$, is defined to be the unique point $y \in G$ that is nearest to x with respect to the Euclidean norm. Using the projection operation, the subgradient algorithm in its most general form follows:

Algorithm 1: Subgradient optimization algorithm

```
0. Initialization.
    Let $y^0 \in G$.
    Select a set of step sizes $s_0, s_1, s_2, \ldots$
    $i \leftarrow 0$.
1. Find subgradient.
    Let $\eta_i \in \delta f(y^i)$.
    If $\eta_i = 0$, then terminate with $y^i$ optimal.
2. Move to new point.
    $y^{i+1} \leftarrow P[y^i - s_i \eta_i]$.
    $i \leftarrow i + 1$, and return to step 1.
```

There are three general schema which can be used in determining the step size when the subgradient algorithm is implemented for a specific problem:

(i) $s_i = \lambda_i$;

(ii) $s_i = \lambda_i / \| \eta_i \|^2$;

(iii) $s_i = \lambda_i (f(y^i) - F) / \| \eta_i \|^2$,

where F is an estimate of f^*, the optimal value of f over G. A summary of the known convergence results for this algorithm may be found in [2] and [10].

3. The lower bound

A lower bound on the objective function of the equal flow problem, I1 or P1, can be obtained by using the Lagrangean dual of the problem. The lower bound is used in the step size determination, termination criteria, and determination of an initial equal flow allocation for the upper bound procedure. Associating the Lagrange multiplier w_k with the k-th equal flow constraint and defining the K-vector $w = (w_1, w_2, \ldots, w_K)$, the Lagrangean dual for P1, referred to as problem D1, may be stated as

$$\operatorname*{Max}_{w \in \mathbb{R}^K} h(w),$$

where $h(w) = \min\{ cx + \sum_k w_k (x_k - x_{K+k}) \mid Ax = b, \mathbf{0} \leqslant x \leqslant u \}$. Since P1 is a linear program, it is easily established that the optimal objective values of P1 and D1 are equal and that any feasible solution to D1 provides a lower bound on the optimal objective value for problems P1 and I1.

For any given value of the vector w, the Lagrangean relaxation is a pure network problem. The subgradient of h at a point $\tilde{w}$ is given by the K-vector

$$d = (\tilde{x}_1 - \tilde{x}_{K+1}, \ldots, \tilde{x}_K - \tilde{x}_{2K})$$

where $\tilde{x}$ solves the Lagrangean relaxation at $\tilde{w}$, given by

$$\{\min cx + \Sigma_k \tilde{w}_k (x_k - x_{K+k}) \mid Ax = b, \; 0 \leqslant x \leqslant u\}.$$

Algorithm 1 assumed the function $f(y)$ to be convex, whereas $h(w)$ is piecewise linear concave. The lower bounding algorithm, Algorithm 2, modifies the framework of the previous algorithm for a concave function. The step sizes used are given by $\lambda_0 = p$, and successive values of λ_i depend on the progressive improvement of the objective and a parameter m^*. As long as the objective function continues to improve across m^* iterations, the same value of the multiplier is retained. If the objective does not improve over m^* iterations, the multiplier is halved, and successive iterations continue from the point where the incumbent best objective function value is found for the previous value of the multiplier. The algorithm makes use of a scalar, UBND, representing an upper bound for the problem. Since the solution procedure progressively improves both the lower bound and the upper bound for the equal flow problem, each time the lower bound algorithm is invoked the value for UBND is obtained from the upper bound procedure. For this algorithm, we assume that both bounds are greater than zero.

Several termination criteria are pertinent to the lower bound algorithm. If the value of the multiplier becomes negligibly small, further improvement in the lower bound is negligible. Such termination criteria are relevant particularly to the initial invocation of the lower bound algorithm since no valid estimate of the upper bound is available. Further, the maximum number of iterations allowed in the initial invocation of the lower bound procedure should be chosen to be larger than in subsequent invocations.

Algorithm 2: Lower bound algorithm

```
1. Initialization.
   Initialize UBND, step size $p$, $m^*$, and tolerance $\varepsilon$.
   $w \leftarrow 0$, $m' \leftarrow 0$, $d' \leftarrow \infty$, $I \leftarrow 0$.
2. Find subgradient.
   $I \leftarrow I + 1$.
   Let $\tilde{x}$ solve $h(w) = \min\{cx + \Sigma_k w_k (x_k - x_{K+k}) \mid Ax = b, 0 \leqslant x \leqslant u\}$.
   $d \leftarrow (\tilde{x}_1 - \tilde{x}_{K+1}, \ldots, \tilde{x}_k - \tilde{x}_{2K})$.
   If $\|d\| < \|d'\|$, $d' \leftarrow d$, $x' \leftarrow \tilde{x}$.
   If $d = 0$, terminate.
   If $h(w) <$ LBND,
      $m' \leftarrow m' + 1$,
      if $m' = m^*$, $p \leftarrow p/2$, $w \leftarrow w^*$, $d \leftarrow d^*$, $m' \leftarrow 0$;
   otherwise,
      $m' \leftarrow 0$,
      LBND $\leftarrow h(w)$,
      $w^* \leftarrow w$,
      $d' \leftarrow d$.
   If (UBND − LBND) $\leqslant \varepsilon$(UNBD), terminate.
3. Move to new point.
   (a) $w \leftarrow w + pd$.
   (b) If $\max\{pd_i\} < 0.005$, terminate.
   (c) Go to step 2.
```

The choice of the initial value of p should be directed by the range of objective function coefficients for the problem as well as an estimate of the elements of the vector d. This choice can be made automatically when the Lagrangean relaxation is solved with $w = 0$. Since it is the elements of d which cause the objective function coefficients to change in each successive iteration of the subgradient optimization procedure, an initial value of p which keeps objective function coefficients from taking on values far away from the original range is a prudent choice. Note that termination of the lower bound procedure can occur when further changes in objective function coefficients is minimal as in step 3(b).

4. The upper bound

An alternate formulation of problem P1, referred to as P2, obtained by decomposing the problem is given by

$$\begin{array}{ll} \text{Min} & g(y), \\ \text{s.t.} & y \in S, \end{array}$$

where for any vector $y = (y_1, y_2, \ldots, y_K)$,

$$g(y) = \{\min \boldsymbol{cx} \mid \boldsymbol{Ax} = \boldsymbol{b};\ \boldsymbol{0} \leqslant \boldsymbol{x} \leqslant \boldsymbol{u}\ \ x_k = x_{K+k} = y_k,\ k = 1, 2, \ldots, K\},$$

and

$$S = \{ y \mid 0 \leqslant y_k \leqslant u_k, \text{ for } k = 1, 2, \ldots, K\}.$$

The decomposition assures the satisfaction of the equal flow constraints. The decomposed problem P2 is equivalent to the problem P1 [12] and may be solved using a specialization of the subgradient optimization algorithm. The objective function is piece-wise linear convex and the subgradient η of g at a point y is obtained from the dual variables, v_i, $i = 1, 2, \ldots, 2K$, associated with the equal flow constraints in the subproblem, referred to as P3 and given by,

$$\begin{array}{lll} \text{Min} & \boldsymbol{cx}, & \\ \text{s.t.} & \boldsymbol{Ax} = \boldsymbol{b}, & \\ & x_1 = y_1, & (v_1) \\ & x_{K+1} = y_1, & (v_{K+1}) \\ & \vdots & \\ & x_K = y_K, & (v_K) \\ & x_{2K} = y_K, & (v_{2K}) \\ & \boldsymbol{0} \leqslant \boldsymbol{x} \leqslant \boldsymbol{u}. & \end{array}$$

The K-vector

$$\eta = (v_1 + v_{K+1}, v_2 + v_{K+2}, \ldots, v_K + v_{2K})$$

is a *subgradient* of g at $y = (y_1, y_2, \ldots, y_K)$.

The dual variables v_k, $k = 1, 2, \ldots, 2K$, may be easily constructed from the solution to the pure network problem, referred to as problem P4:

$$\{\min \boldsymbol{cx} \mid \boldsymbol{Ax} = \boldsymbol{b},\ \boldsymbol{\gamma} \leqslant \boldsymbol{x} \leqslant \boldsymbol{\theta}\},$$

where the lower and upper bound n-vectors $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ are defined by

$$\begin{array}{ll} \gamma_k = \theta_k = y_k, & k = 1, 2, \ldots, K, \\ \gamma_{K+k} = \theta_{K+k} = y_k, & k = 1, 2, \ldots, K, \\ \gamma_k = 0,\quad \theta_k = u_k, & k = 2K+1, \ldots, n. \end{array}$$

Let Π be the vector of optimal dual variables associated with the conversion flow constraints, $\mathrm{A}\boldsymbol{x} = \boldsymbol{b}$ in P4 and the arc associated with the variable x_j be incident from node j_f and incident to node j_t. The optimal dual variables for P3 are given by

$$v_k = -\Pi_{k_f} + \Pi_{k_t} + c_k, \quad k = 1, 2, \ldots, 2K.$$

In using the subgradient optimization algorithm for the decomposed problem at each point y, the subgradient η can be calculated directly using the above development.

It is possible that moving a step along the negative subgradient yields a point which does not belong to the set S. As pointed out in Section 2, this point is projected onto the set S by means of a projection operation in the algorithm. For this model, the projection operation decomposes on k so that $P[y] = (P[y_1], P[y_2], \ldots, P[y_K])$, where the projections $P[y_k]$ are defined by

$$\begin{array}{ll} P[y_k] = 0 & \text{if } y_k < 0, \\ P[y_k] = u_k & \text{if } y_k > u_k, \\ P[y_k] = y_k & \text{if } 0 \leqslant y_k \leqslant u_k. \end{array}$$

The subgradient optimization algorithm for problem P2 makes use of a lower bound, LBND, on the optimal objective value which is used in step size determination using a variant of scheme (iii) given in Section 2, as well as in the termination criteria. Again, we assume that both bounds are greater than zero.

Algorithm 3: Upper bound algorithm

```
1. Initialization.
   Select $y \in S$ and construct $\gamma$ and $\theta$.
   Initialize LBND, $\varepsilon$, $q$, $n^*, J \leftarrow 0$.
2. Find subgradient and step size.
   $J \leftarrow J + 1$.
   Let $\tilde{x}$ and $\Pi$ be the vectors of optimal primal
      and dual variables for $\min\{cx \mid Ax = b,\ \gamma \leqslant x \leqslant \theta\}$.
   If $c\tilde{x} >$ UBND,
      $n' \leftarrow n' + 1$,
      if $n' = n^*$, $q \leftarrow q/2$, $n' \leftarrow 0$;
   otherwise,
      $n' \leftarrow 0$,
      UBND $\leftarrow c\tilde{x}$.
   If (UNBD − LBND) $\leqslant \varepsilon$(UBND) and $\tilde{x}$ feasible,
      terminate with $\tilde{x}$ optimal; otherwise,
      $v_k \leftarrow -\Pi_{k_f} + \Pi_{k_t} + c_k$, $k = 1, 2, \ldots, 2K$.
      $\eta \leftarrow (v_1 + v_{K+1}, \ldots, v_K + v_{2K})$.
3. Move to new point.
   (a) $y \leftarrow P[y - q((\text{UBND} - \text{LBND})/(\|\eta\|^2))\eta]$.
   (b) If $\max(q((\text{UBND} - \text{LBND})/(\|\eta\|^2))\eta])_i <$
   0.01 then terminate.
   (c) Go to 2.
```

The use of the algorithm parameters q and n^* is to help condition the step sizes based on the relative norm of the subgradient with respect to the difference in the lower and upper bounds. The norm of the subgradient is dependent on the problem rather than the algorithm. That is, it is quite possible that the norm remains high throughout the algorithm. The initial choice of q is directed by an estimate of the maximum of the absolute values of the elements of the vector $\boldsymbol{d}'$ as well as the objective function coefficient associated with artificial variables in the solution of the pure network problem. When allocations yield infeasible solutions, the elements of $\boldsymbol{\eta}$ are large rendering $\|\boldsymbol{\eta}\|^2$ very large. An initial value of q, if chosen arbitrarily, can be small, thus requiring more iterations since the improvement at each iteration is small. On the other hand, if the initial value of q is large, then for several sets of n^* iterations no improvement in the objective occurs until the value of q becomes smaller. Here again, a secondary termination criterion in step 3(b) is when further changes in the allocation in step 3(a) are minimal.

The modification required for the integer problem occurs only when the termination criteria have been met for the linear problem. The alternate formulation for the integer problem is obtained by requiring that the equal flow allocation, y be integral:

$$\begin{aligned}&\text{Min} \quad g(y),\\ &\text{s.t.} \quad y \in S^1,\end{aligned}$$

where for any vector $y = (y_1, y_2, \ldots, y_K)$,

$$g(y) = \{\min \boldsymbol{cx} \mid A\boldsymbol{x} = \boldsymbol{b};\ \boldsymbol{0} \leqslant \boldsymbol{x} \leqslant \boldsymbol{u};\ x_k = x_{K+k} = y_k,\ k = 1, 2, \ldots, K\},$$

and

$$S^1 = \{y \mid 0 \leqslant y_k \leqslant u_k,\ \text{for } k = 1, 2, \ldots, K \text{ and } y \text{ integer}\}.$$

Once termination occurs for the linear problem, the upper bound algorithm can be used by requiring that the projection in step 3 of the algorithm yield an integer equal flow allocation. Since the objective retains the piece-wise convex nature of the objective to the linear problem, the linear optimum obtained can be expected to be close to the integer optimum. Adjacent integer allocations can be expected to provide bounds on the integer optimum or else be near-feasible points for the integer problem.

5. The algorithm

The solution of the equal flow problem using decomposition, as given in the previous section can be implemented without the lower bound procedure. It is also possible to implement the lower bound algorithm independently for the purpose of obtaining a lower bound on the optimal value of the equal flow problem. For the upper bound problem, some measure of the lower bound on the problem must be used to aid in termination. By merging the two procedures, an algorithm which adjusts the lower and upper bounds progressively can be used to advantage and tied to the accuracy desired for the solution. Not only can such a procedure be used for obtaining feasible solutions with relative ease, but it can also provide a measure of how close this solution is to the optimal.

The algorithm for the solution of the equal flow problem iterates between the lower bound procedure and the upper bound procedure. The lower and upper bounds, LBND and UBND, progressively become tighter, closing in on the optimal solution to the problem. Each time the lower bound procedure is invoked, a maximum of ITERL iterations are performed. Each time the upper bound procedure is invoked, a maximum of ITERU iterations are performed. However, the initial invocations of the lower and upper bound algorithms are allowed to terminate using criteria in those algorithms as opposed to these iteration counts. The initial invocation of the Lagrangian dual is important primarily because it affords a very tight lower bound on the objective value of the integer or linear optimum and further it provides near-optimal values of the Lagrange multipliers. The near-optimal values of the Lagrange multipliers tend to aid the subgradient optimization of the decomposition model. The tuning parameters for the algorithm are as follows: ITERL, ITERU, m', n', and ε (the termination criterion.)

Algorithm 4: Relaxation/decomposition algorithm for the equal flow problem

```
0. Initialization.
    Initialize ITERL, ITERU, $\varepsilon$.
    $T \leftarrow 0$, $R \leftarrow 0$, $\boldsymbol{w} \leftarrow \boldsymbol{0}$, UBND $\leftarrow \infty$, LBND $\leftarrow -\infty$.
    Call Algorithm 2 and
    $y_k \leftarrow \min[u_k, (x'_k + x'_{K+k})/2]$, $k = 1, 2, \ldots, K$.
    Call Algorithm 3.
```

```
1. Compute lower bounds.
   (a) Call Algorithm 2 (Steps 2 and 3 (a)).
   (b) $T \leftarrow T + 1$.
       If $T <$ ITERL, then go to step 1 (a).
2. Compute upper bounds.
   (a) Call Algorithm 3 (Steps 2 and 3 (a)).
   (b) $R \leftarrow R + 1$.
       If R < ITERU, then go to step 2 (a).
3. Reset iteration counts.
   $T \leftarrow 0$, $R \leftarrow 0$, and go to step 1.
```

6. Computational experience

The computer implementation of the algorithm is written in standard FORTRAN (called EQFLO) and makes use of MODFLO [1] to solve pure network subproblems. Based on NETFLO [8], MODFLO is a set of subroutines which allows parametric changes in costs, bounds and/or requirements for a network problem and subsequent reoptimization. Computational testing was carried out on the IBM 3081D at The University of Texas at Austin using the FORTVS compiler with OPT = 2. In order to assess the computational gains afforded by the decomposition/relaxation algorithm for the equal flow problem, each problem was solved using MPSX [7]. All MPSX solutions have been obtained on the IBM 3081D at Southern Methodist University.

The algorithm has been tested on a set of 10 test problems generated by using NETGEN [9], and referred to by their NETGEN numbers. Of the 10 problems used, the first three are transportation problems (problems 5, 9, and 10), the next four are capacitated transshipment problems (problems 20, 21, 24, and 25) and the last three are uncapacitated transshipment problems (problems 28, 30, and 35). The test problems have between 200 and 1500 nodes, and between 1500 and 6600 arcs. For each problem, the first $2K$ arcs were paired to form K equal flow side constraints. In order to gauge the performance of the algorithm for various values of K, some of the problems were generated using the same base network problem data with K varying from 75 to 200.

The benchmark NETGEN problems have a specified percentage of arcs which are uncapacitated. For these arcs, the capacity was defined to be the maximum of all supplies and demands. For arcs in equal flow pairs which emanate from supply points, the capacity used is the supply at the point of incidence. Similarly, for arcs incident to demand points, the capacity used is the corresponding demand. If an equal flow pair is incident to a demand point or incident from a supply point, then the capacity assigned is the upper ceiling of half the corresponding requirement. Such allocation of capacity is prudent, allowing a tighter relaxation.

Table 1 details the computational testing of the algorithm with parameters $m^* = 5$, $n^* = 10$, ITERU = ITERL = 10, $\varepsilon = 0.01$. For the test problems, EQFLO obtained feasible solutions whose objective function values were within 1% of the optimal in a fraction of the time required by MPSX to obtain an optimum. The table reports the total solution times required to produce an α percent

Table 1
Comparison of EQFLO with MPSX (all problems have 75 equal flow pairs)

NETGEN			MPSX	Linear					Integer	
Number	Nodes	Arcs	Time	α	$\|d'\|$	I	J	Time	Infeas	Time
5	200	3100	11.4	0.15	377	151	1	8.6	4	8.8
9	300	6395	38.4	0.01	1296	145	1	19.6	3	19.9
10	300	6611	34.2	0.00	698	165	1	15.3	3	15.7
20	400	1484	34.8	0.10	4729	544	262	23.3	2	23.6
21	400	2904	73.8	0.00	1	6	1	00.7	1	1.1
24	400	1398	37.8	0.29	8875	280	498	21.4	3	21.7
25	400	2692	93.0	0.01	8356	148	1	5.7	1	6.3
28	1000	3000	52.8	0.14	3865	235	220	27.8	29	28.3
30	1000	4500	69.6	0.00	490	95	1	7.9	0	7.9
35	1500	5880	145.2	0.09	5386	218	151	46.7	6	47.7
			591.0					177.0		181.0

Times reported are in CPU seconds on an IBM308ID.

solution for the linear equal flow problem and an integer solution. The number of lower and upper bound iterations, respectively I and J, required are provided along with the norm, $\|d'\|$, obtained during subgradient optimization of the Lagrangean dual. Note that this norm typically provides a metric for gauging the difficulty of a specific problem instance. Because of the fact that the lower bound procedure does not enforce equal flow, the norm provides a measure of the infeasibility of equal flow constraints, or the amount of imbalance which exists in the problem. The zero-tolerance used for flows on artificial arcs is 0.05. Of the 10 problems, feasible solutions were obtained for all linear problems. Termination criteria employed for this computational testing are stringent and the decomposition algorithm continues to attempt improvement until not only the solution is within 1% of the lower bound, but also until changes in subsequent allocations are negligible (0.001).

Initial allocations, as determined by x', obtain feasible solutions well within 1% of the lower bound obtained in 6 of the 10 problems. For the other problems, the initial allocation can be feasible or infeasible. Capacities in the randomly generated problems are such that infeasibility occurs due to the following: When a particular level of allocation is enforced, the problem can become infeasible due to capacities falling below a level required to ensure all demand be met.

An upper limit of 5 iterations were allowed in performing integer equal flow allocations with the initial integer allocation obtained by truncating the optimal linear allocation. No more than 29 units of demand go unsatisfied corresponding to 99.99% feasible integer solutions well within 1% of the lower bound obtained. The trade-off between enforcing integer equal flows and 100% feasible solutions tips in favour of making use of near-feasible solutions, given the relative computational ease with which they are produced. Problem 5 was attempted with MPSX-MIP where integrality was only forced on the 75 equal flow pairs. After over 220 seconds the active branch-and-bound tree had over 2000 nodes and had not as yet obtained the first feasible integer solution. In less than 9 seconds, the decomposition procedure obtained an integer solution which satisfied 399 996 units of the 400 000 units of demand.

To determine the impact of an increase in the number of side constraints on problem characteristics and the algorithm, additional testing with 21, 24, and 28 is reported in Table 2. Each of these base problems was used to generate equal flow problems with 75, 100, 150, and 200 equal flow constraints. As evident from the behavior of the norm of d', as the number of side constraints increases, more imbalance in the problems is introduced and in order to enforce equal flow, more effort is required. Problem 24 becomes infeasible once the number of side constraints enforced becomes 200. As would be expected, the algorithm expends more effort for the more tightly constrained problems with the exception that it recognizes an feasible problem readily. Again, for the

Table 2
Effect of increasing the number of equal flow pairs

NETGEN		MPSX	Linear					Integer	
Number	Pairs	Time	α	$\|d'\|$	I	J	Time	Infeas	Time
21	75	73.8	0.00	1	6	1	00.7	1	1.1
21	100	64.8	0.00	1	6	1	00.8	1	1.1
21	150	83.4	0.08	880	130	202	14.6	2	15.1
21	200	76.2	0.47	2937	214	188	17.5	26	17.9
24	75	37.8	0.29	8875	280	498	21.4	3	21.7
24	100	36.0	0.53	18283	240	307	19.8	5	19.9
24	150	42.0	2.32	24867	258	505	30.7	37	30.9
24	200						infeasible		infeasible
28	75	52.8	0.14	3865	233	220	27.8	29	28.3
28	100	57.6	0.23	5206	213	182	28.3	35	28.9
28	150	65.4	0.30	6043	202	179	30.4	45	30.9
28	200	72.0	1.00	15206	224	423	50.5	49	51.0
		661.8					242.5		246.8

Times reported are in CPU seconds on an IBM3081D.

problems which are feasible, near-feasible integer solutions are obtained in approximately 1%–60% of the time required to solve the linear problem using MPSX.

7. Summary and conclusions

The equal flow problem lends itself to solution by decomposition and relaxation. The use of these techniques in the solution procedure developed is advantageous because the essential solution mechanism required is the solution of sequences of pure network problems. By dispending with the working basis required by other techniques, not only are computational efficiencies afforded but the natural characteristics of the problem enhanced.

The algorithm has been shown to assist in solving the integer equal flow problem. The lower bound automatically produces integer flows and the projection of the subgradient in the upper bound routine is altered to require integrality on the equal flow allocation once a near-optimal linear solution has been obtained. The equal flow allocation for the linear model is expected to be close to the equal flow allocation for the integer model due to problem structure. Thus the solution procedure provides near-feasible, near-optimal solutions for the integer equal flow problem efficiently.

The structure of the equal flow problem provides a metric on the relative difficulty of any specific problem instance. The proposed solution procedure has the innate capability to distinguish between easy and difficult instances of an equal flow problem and thus can require only 1% of the MPSX time to solve an easy problem. As the number of equal flow constraints grows, a problem can become progressively infeasible, since the enforcement of equal flows can serve to reduce the capacity of a cut-set of the network to well below required levels for feasibility. The development for the linear equal flow problem in this paper can be instructive in modelling and solving other network problems with specially structured side constraints such as proportional flow models used in manpower planning. The solution technique is best suited for a real-world situation in which one must quickly produce near-optimal, near-feasible solutions.

References

[1] Ali, A., Allen, E., Barr, R., and Kennington, J., "Reoptimization procedures for bounded variable primal simplex network algorithms", *European Journal of Operational Research* 23 (1986) 256–263.

[2] Allen, E., Helgason, R., Kennington, J., and Shetty, B., "A generalization of Polyak's convergence result for subgradient optimization", Technical Report 85-OR-7, Department of Operations Research, Southern Methodist University, Dallas, TX 75275, 1985, to appear in *Mathematical Programming*.

[3] Barr, R., Farhangian, K., and Kennington, J., "Networks with side constraints: An LU factorization update", *The Annals of the Society of Logistics Engineers* 1/1 (1986) 66–85.

[4] Beck, P., Lasdon, L., and Engquist, M., "A reduced gradient algorithm for nonlinear network problems", *ACM Transactions on Mathematical Software* 9 (1983) 57–70.

[5] Carraresi, P., and Gallo, G., "Network models for vehicle and crew scheduling", *European Journal of Operational Research* 16 (1984) 139–151.

[6] Glover, F., Glover, R., and Martinson, F., "The U.S. Bureau of Land Management's new NETFORM vegetation allocation system", Technical Report of the Division of Information Science Research, University of Colorado, Boulder, Colorado, 80309, 1982.

[7] IBM Mathematical Programming System Extended/370 Program Reference Manual, File No. S370-82, IBM Corp., White Plains, NY, 1979.

[8] Kennington, J., and Helgason, R., *Algorithms for Network Programming*, Wiley, New York, 1980.

[9] Klingman, D., Napier, A., and Stutz, J., "NETGEN: A program for generating large scale minimum cost flow network problems", *Management Science* 20 (1974) 814–821.

[10] Poljak, B.T., "A general method of solving extremum problems", *Soviet Mathematics Doklady* 8/3 (1967) 593–597.

[11] Shepardson, F., and Marsten, R., "A Lagrangian relaxation algorithm for the two duty period scheduling problem", *Management Science* 26 (1980) 2274–281.

[12] Shetty, B., "The equal flow problem", unpublished dissertation, Department of Operations Research, Southern Methodist University, Dallas, TX, 75275, 1985.

[13] Shor, N., "On the structure of algorithms for the numerical solution of optimal planning and design problems", Dissertation, Cybernetics Institute, Academy of Sciences, U.S.S.R., 1964.

[14] Turnquist, M., and Malandraki, C., "Estimating driver costs for transit operations planning", Joint National Meeting of ORSA/TIMS, Dallas, 1984.

[15] Zangwill, W., *Nonlinear Programming: A Unified Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1969.