OPERATIONS RESEARCH CENTER

working paper

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

Tailoring Benders Decomposition
For Uncapacitated Network Design

Thomas L. Magnanti
Richard T. Wong
Paul Mireault

OR 127-84 September 1984

ABSTRACT

Because of its imbedded network flow structure, the generic network design problem is an attractive candidate for integer programming decomposition. This paper studies the application and acceleration of Benders decomposition for uncapacitated models from this problem class and illustrates the potential flexibility of the Benders solution strategy. In particular, it (i) shows that several lower bounding inequalities from the literature can be derived as Benders cuts; and (ii) introduces new Benders cuts for the network design problem.

The paper also reports on computational experience in using Benders decomposition with a dual ascent and variable elimination preprocessing procedure to solve uncapacitated network design problem with up to 90 binary variables and 15,080 continuous variables, or 45 binary variables and 105,600 continuous variables.

KEY WORDS: Network Design, Benders Decomposition, Dual Ascent, Variable Elimination, Integer Programming.

ABBREVIATED TITLE: Benders Decomposition for Network Design

INTRODUCTION

Conceptually, integer programming models of network design are very simple. They address the following basic question: what configuration of the network minimizes the sum of the fixed costs of arcs chosen to be in the network (a set of binary decisions) and the costs of routing goods through the network defined by these arcs? And yet, when specialized, even the simplest version of this problem with linear routing costs and without arc capacities models many of the most well-known problems in combinatorial optimization. These special cases include shortest path problems, minimal spanning tree problems, optimal branching problems, and the celebrated traveling salesman problem and network Steiner problem (Magnanti and Wong [1984a]).

Moreover, as illustrated by several books and edited collections of papers (e.g., Boesch [1975], Boyce [1979], and Mandl [1981]), the generic network design problem has numerous applications. Not only does it model a variety of traditional problems in communication, transportation, and water resource planning, but it also arises in emerging problem contexts such as flexible and automated manufacturing systems (Graves and Lamar [1983] and Kimenia and Gershwin [1979]). In addition, familiar node splitting devices permit the model to represent facility location decisions that have applications in warehouse and plant location, emergency facility location, computer networking, and many other problem domains. (See Francis and White [1974], Handler and Mirchandani [1979], Larson and Odoni [1981], and Tansel, Francis and Lowe [1983] for extensive citations to this literature.) Finally, the network design problem arises in many vehicle fleet planning applications that do not involve the construction of physical facilities, but rather model service decisions, i.e., do we dispatch a vehicle on a link or not (e.g., see Simpson [1969] or Magnanti [1981]).

Although these applications have attracted numerous algorithmic studies, even the uncapacitated network design problem remains essentially unsolved. Surprisingly, the largest problems solved to optimality (and verified as such) contain only about 10 nodes and 40-50 arcs. This experience is, in part, explainable theoretically. Not only are the simplest versions of the network design problem NP-complete (Johnson, Lenstra and Rinnooy Kan [1978], Wong [1978]), but finding even approximate solutions (to problems with budget restrictions on the fixed costs incurred) is NP-complete (Wong [1980]). Even so, there does not appear to be any compelling evidence to explain why network design remains among the most computationally elusive of all integer programs encountered in practice.

In this paper, we study Benders decomposition-based algorithms for the uncapacitated network design problem. Although Benders decomposition has proved to be a useful planning tool in several problem contexts including distribution system design (Geoffrion and Graves [1974]), aircraft routing (Richardson [1976]), and rail engine scheduling (Florian, Guerin and Bushel [1976]), its successes remain limited. We nevertheless feel that the method continues to have great potential. By studying the algorithm as applied to the uncapacitated network design problem and using it together with other integer programming solution techniques, we

(i) report on computational results on some of the largest network design problem solved to optimality to date. The problems studied in this paper contain up to 90 binary variables and 15,080 continuous variables or 45 binary variables and 105,600 continuous variables (more compact models with as few as 480 and 6,600 continuous variables are possible, though these alternate formulations are much less attractive computationally);

(ii) adapt and enhance algorithmic ideas from Magnanti and Wong [1981, 1983b] that demonstrate the potential for accelerating Benders decomposition and for generating a rich variety of lower bound inequalities for the network design problem and other mixed integer programs. In particular, we show that several lower bounding procedures for the network design problem from the literature, even though originally derived by other means, can be viewed as special instances of Benders cuts; and

(iii) illustrate the importance of using other solution procedures (bounding-based variable elimination methods, dual ascent) together with Benders decomposition as part of a comprehensive approach to solving integer programming problems.

Previously, researchers have attempted to solve uncapacitated network design problems using a variety of solution techniques. Hoang [1973], Boyce, Farhi and Weischedel [1973], Dionne and Florian [1979], Boffey and Hinxman [1979], and Los and Lardinois [1982] have all studied branch and bound algorithms for the problem. Gallo [1981] has proposed a branch and bound procedure for the related "optimal network design problem" that eliminates the fixed costs from the objective function, and instead limits fixed cost expenditures by imposing a given budget as a constraint. Although these branch and bound algorithms can successfully solve problems with a small number (in some cases up to 40-50) of arcs, their computation time grows very quickly in the problem size.

Heuristic procedures are capable of generating solutions to much larger network design problems. Billheimer and Gray [1973] describe an add-drop heuristic method with provisions for eliminating non-optimal variables. Los and Lardinois [1982] suggest improvements to this method and discuss statistical methods for analyzing the solutions that it generates. Dionne and Florian [1979] and Boffey and Hinxman [1979] propose heuristics for the optimal network design problem. Wong [1984] gives a special heuristic for optimal network problems on the Euclidean plane. He shows that with high probability, the cost of the solution generated by this heuristic will, under certain conditions, be very close to the cost of the optimal solution.

These various heuristics can usually solve larger-sized problems (20-50 nodes) in a small amount of computation time. However, it is usually difficult to assess the quality of the heuristics' solutions since no satisfactory method is known for solving problems of this size optimally.

In one study of integer programming decomposition for the network design problem, Rardin and Choe [1979] have devised a Lagrangian relaxation algorithm. Their computational results seem to be quite promising.

Our computational results in this paper demonstrate that choosing Benders cuts judiciously can have a marked effect on the algorithm's performance. We introduce a new type of cut, which is generated by using information from solving a single shortest-path problem over all candidate arcs at the outset of the computations. This new cut and a Pareto-optimal cut generated by specializing the methodology from Magnanti and Wong [1981] are an order of magnitude more effective than cuts generated by the usual implementation of Benders method.

Our computational experience also indicates that Benders decomposition can be much more effective when used in conjunction with other integer programming techniques (dual ascent and variable elimination procedures). Equipped with these enhancements, the algorithm was able to solve to optimality 19 of 24 test problems with 45 arcs (in three cases) and 90 candidate arcs, and to find solutions to 23 of these problems that are guaranteed to be within at most 1.44% of optimality (and to within at most 5.53% for the 24th problem). The computations took from about one minute to about 1-1/2 hours on a VAX 11/780 computer, which we estimate would correspond to about 7 seconds to 10 minutes on an IBM 3033. These solution times were obtained using the branch and bound facilities in the Land and Powell [1973] mathematical programming system. They could be reduced, and quite likely significantly, by using a commercial branch and bound code or specialized computer codes for solving the Benders master integer program.

This experience indicates that large-scale uncapacitated network design problems are within the reach of current integer programming capabilities; moreover, it highlights the importance of adopting a holistic view of integer programming methods, rather than treating each solution procedure in isolation.

1. MODELING AND SOLUTION APPROACH

Problem Formulation

The basic ingredients of the model are a set N of nodes and a set A of undirected arcs that are available for designing a network.

The model has multiple commodities. These might represent distinct physical goods, or the same physical good but with different points of origin and destination. We let K denote the set of commodities and for each $k \in K$, assume (by scaling, if necessary) that one unit of flow of commodity k must be shipped from its point of origin, denoted O(k), to its point of destination denoted D(k). This formulation can model problems in which any commodity has either multiple origins or multiple destinations by considering the flow from each origin to each destination as a separate commodity (i.e., by disaggregating flow). The formulation cannot, however, model problems with both multiple origins and multiple destinations since it then loses its imbedded shortest path structure.

The model contains two types of variables, one modeling discrete choice design decisions and the other modeling continuous flow decisions. Let y_{ij}

be a binary variable that indicates whether ($y_{ij} = 1$) or not ($y_{ij} = 0$) arc $\{i,j\}$ is chosen as part of the network's design. Let x^k_{ij} denote the flow of commodity k on arc (i,j). Note that (i,j) and (j,i) denote directed arcs corresponding to the undirected arc $\{i,j\}$. Even though arcs in the model are undirected, we refer to these directed arcs because the flows are directed. Then, if $y = (y_{ij})$ and $x = (x^k_{ij})$ are vectors of design and flow variables, the model becomes:

$$P_1: \quad \text{: minimize} \sum_{k\in K} \sum_{\{i,j\}\in A} (c^k_{ij} x^k_{ij} + c^k_{ji} x^k_{ji}) + \sum_{\{i,j\}\in A} F_{ij} y_{ij} \qquad (1.1)$$

subject to:

$$\sum_{i\in N} x^k_{ij} - \sum_{\ell\in N} x^k_{j\ell} = \begin{cases} -1 \text{ if } j = 0(k) \\ 1 \text{ if } j = D(k) \\ 0 \text{ otherwise} \end{cases} \qquad \text{all } j\in N, \text{ all } k\in K \qquad (1.2)$$

$$\left.\begin{array}{l} x^k_{ij} \le y_{ij} \\ x^k_{ji} \le y_{ij} \end{array}\right\} \qquad \text{all } \{i,j\}\in A, \text{ all } k\in K \qquad (1.3)$$

$$x^k_{ij}, x^k_{ji} \ge 0, \ y_{ij} = 0 \text{ or } 1 \qquad \text{all } k\in K, \{i,j\}\in A \qquad (1.4)$$

$$y\in Y. \qquad (1.5)$$

In this formulation c^k_{ij} is the nonnegative per unit cost for routing commodity k on arc (i,j) and F_{ij} is the fixed charge design cost for arc $\{i,j\}$. In general, c^k_{ij} need not equal c^k_{ji}. Constraints (1.2) imposed upon each commodity k are the usual network flow conservation equations (multiplied through by -1 to facilitate later interpretations of our algorithmic development). The "forcing" constraints (1.3) state that if arc $\{i,j\}$ is included in the design, i.e., if $y_{ij} = 1$, then the total arc flow is unlimited (since the flow of any commodity k on arc (i,j) or (j,i) is at most one, anyway) and if arc $\{i,j\}$ is not included in the network design, i.e., if $y_{ij} = 0$, then the total arc flow must be zero. Finally, the set Y imposes any possible restrictions on the design variables such as multiple choice constraints ($y_{ij} + y_{pq} + y_{rs} \leq 1$) or precedence constraints ($y_{ij} \leq y_{rs}$).

Like most integer programming problems, this design problem could be modeled in a variety of ways. In particular, the forcing constraints (1.3) are equivalent to the more aggregated constraints

$$\left.\begin{array}{l} \sum_{k\epsilon K} x^k_{ij} \leq |K| y_{ij} \\ \text{and} \\ \sum_{k\epsilon K} x^k_{ji} \leq |K| y_{ij} \end{array}\right\} \quad \text{for all } \{i,j\}\epsilon A. \tag{1.6}$$

Both versions of these constraints force each x^k_{ij} and x^k_{ji} for $k\epsilon K$ to be zero if $y_{ij} = 0$, and become redundant if $y_{ij} = 1$.

This aggregation could substantially reduce the number of constraints in the formulation for even moderately sized problems. With 50 commodities and 100 arcs, the disaggregate formulation (1.3) contains 10,000 forcing constraints; the aggregate version contains only 200 forcing constraints.

Note that the formulation (1.1)-(1.5) includes a single commodity k for each origin-destination pair of demand. This definition of commodities gives rise to 2 constraints of the form (1.3) for each origin-destination pair. Alternatively, when the per unit routing costs are independent of the destination, we could distinguish commodities only by their point of origin (then x^k_{ij} is the total flow from origin k on arc (i,j)), and model the forcing constraints as

$$x^k_{ij} \leq |N| y_{ij}$$

$$x^k_{ji} \leq |N| y_{ij} \quad .$$

In a problem with as few as two origins shipping goods to 25 destination nodes over 100 candidate arcs, this modified model would reduce the number of continuous variables from $2 \cdot 25 \cdot 2 \cdot 100 = 10{,}000$ to $2 \cdot 2 \cdot 100 = 400$.

Surprisingly, the seemingly less efficient and substantially larger disaggregate formulation with forcing constraints (1.3) instead of (1.6), and with flow variables for each origin-destination rather than for each origin, leads to more efficient algorithms. The disaggregated model is preferred computationally for two reasons. First, many techniques for solving integer programs like the network design problem first solve the linear programming relaxation of the model that results by replacing $y_{ij} = 0$ or 1 with $0 \leq y_{ij} \leq 1$. Because the linear programming version of the disaggregate formulation is much more tightly constrained than the linear programming version of the aggregate formulation, the disaggregate linear program provides a sharper lower bound on the value of the integer programming formulation; that is, it more closely approximates the integer program. A number of authors have noted the important advantages of using "tight" linear programming relaxations. (See, for example, Cornuejols, Fisher and Nemhauser [1977], Davis and Ray [1969], Beale and Tomlin [1972], Geoffrion and Graves [1974], Mairs et al. [1978], Magnanti and Wong [1981], Rardin and Choe [1979], and Williams [1974]).

In addition, solution methods such as Benders decomposition that utilize information obtained from the dual of the linear programming relaxation will also be much more effective when applied to the disaggregate formulation (1.1)-(1.5). Because the disaggregate linear programming formulation has more constraints, it has a richer collection of linear programming dual variables and, therefore, provides more flexibility in algorithmic development. Indeed, these two advantages of the disaggregate formulation are intimately related (Magnanti and Wong [1981]).

Benders Decomposition

Benders decomposition assumes a particularly simple form when applied to the uncapacitated network design problem defined by problem P_1. The algorithm alternately solves for a tentative network configuration in the integer variables y and a routing problem in the continuous variables x. For any particular choice of the y variables, the design model (1.1)-(1.5) reduces to a collection of independent shortest path problems, one for each commodity k. Since the mechanics of Benders decomposition is the same for one commodity as it is for many, while describing the algorithm we drop the index k and assume that there is only one commodity (and hence only one routing subproblem) for any fixed value of y. Let S(y) denote this routing subproblem and let R(y) denote its optimal objective value (ignoring the fixed costs). In addition, let u_i and v_{ij} (v_{ji}) denote any feasible dual variables corresponding to the constraints (1.2) and the negative of constraints (1.3), respectively, and suppose that node 1 is the origin node for this single commodity and that node n is its destination node.

By linear programming duality theory, for any fixed choice of y, the value of the dual objective function to the subproblem S(y) is a lower bound on R(y). That is,

$$R(y) \geq u_n - u_1 - \sum_{\{i,j\}\epsilon A} (v_{ij} + v_{ji})y_{ij}.$$

Moreover, this inequality is satisfied as an equality if the variables u_i and v_{ij} solve the dual problem. Therefore,

$$R(y) = \text{minimum } w$$

subject to: (1.7)

$$w \geq u_n - u_1 - \sum_{\{i,j\}\epsilon A} (v_{ij} + v_{ji})y_{ij} \quad \text{for all } (u,v)\epsilon\ \mathcal{D}.$$

In this formulation, $u = (u_i)$ and $v = (v_{ij})$ are vectors of dual variables and $\mathcal{D}$ denotes the dual feasible region for the subproblem formulated with constraints (1.3) multiplied by minus one. This convention of multiplying by minus one ensures that the variables v_{ij} and v_{ji} in $\mathcal{D}$ are nonnegative.

Since the optimal value C* of the design problem equals the minimum of the fixed costs $\sum_{\{i,j\}\varepsilon A} F_{ij}y_{ij}$ and routing costs R(y) over all feasible network configurations,

$$C^* = \text{minimum } z$$

subject to:

$$z \geq \sum_{\{i,j\}\varepsilon A} F_{ij}y_{ij} + u_n - u_1 - \sum_{\{i,j\}\varepsilon A} (v_{ij} + v_{ji})y_{ij} \quad \text{for all } (u,v)\varepsilon \mathcal{D} \quad (1.8)$$

$$y\varepsilon Y \text{ and } y_{ij} = 0 \text{ or } 1 \qquad \text{for all } \{i,j\}\varepsilon A.$$

This reformulation of the original problem is known as the master problem. Note that z and the y_{ij}'s are its decision variables.

Benders decomposition solves the master problem by relaxing the inequality constraints (1.8) on z for most $(u,v)\varepsilon\mathcal{D}$ and replacing them with a finite number of constraints obtained by restricting the (u,v)'s to some small set $\mathcal{D}' \subseteq \mathcal{D}$. Since the constraints are relaxed, the value C' of the relaxed master problem is a lower bound on C*. The algorithm then checks to see if the solution C'=z' and $y' = (y'_{ij})$ to the relaxed master problem is feasible in the full problem by solving the routing subproblem; that is, by solving the shortest path problem defined by y' or its dual reformulation (1.7) with y=y'. If the optimal dual variables (u', v') to this problem satisfy (1.8), i.e., if

$$z' \geq \sum_{\{i,j\}\varepsilon A} F_{ij}y'_{ij} + u'_n - u'_1 - \sum_{\{i,j\}\varepsilon A} (v'_{ij} + v'_{ji})y'_{ij}, \quad (1.9)$$

then (1.8) is satisfied for every other $(u,v)\varepsilon\mathcal{D}$ and thus C* = C', and y' is an optimal configuration. Otherwise, we add the Benders cut (1.9) to the restricted master problem and repeat the procedure.

This procedure requires one further elaboration. For some values $\bar{y}$ of y, the subproblem might be infeasible. That is, the network defined by $\bar{y}$ might not contain any path from the commodity's origin to its destination. In this case, there must be a cut set C separating the origin and destination with $\bar{y}_{ij} = 0$ for all arc $\{i,j\} \varepsilon C$. When this occurs, we add the feasibility cut

$$\sum_{\{i,j\} \varepsilon C} y_{ij} \geq 1$$

to the relaxed master problem and proceed as before. (The full master problem would contain all cuts of this nature.)

For multiple commodity formulations, we let $S_k(\bar{y})$ denote the routing subproblem for commodity k for a fixed value $\bar{y}$ of y, let $R_k(\bar{y})$ denote its optimal objective value, and let $R(\bar{y}) = \Sigma_{k \varepsilon K} R_k(\bar{y})$. Also, let $u^k = (u_i^k)$ and $v^k = (v_{ij}^k)$ denote dual variables for the kth subproblem. Then the method is much the same, except that the Benders cuts in (1.8) becomes

$$z \geq \sum_{\{i,j\} \varepsilon A} F_{ij} y_{ij} + \sum_{k \varepsilon K} [(u_{D(k)}^k - u_{O(k)}^k) - \sum_{\{i,j\} \varepsilon A} (v_{ij}^k + v_{ji}^k) y_{ij}]. \quad (1.10)$$

This cut can be interpreted as follows. Let u_i^k denote the length of the shortest path from node O(k) to node i with respect to distances c_{ij}^k and c_{ji}^k in the network defined by those arcs $\{i,j\} \varepsilon A$ with $\bar{y}_{ij} = 1$. Furthermore, let

$$v_{ij}^k = \max \{u_j^k - u_i^k - c_{ij}^k, 0\} \quad (1.11)$$

$$\text{and} \quad v_{ji}^k = \max \{u_i^k - u_j^k - c_{ji}^k, 0\} .$$

Then $\{u_i^k\}$ and $\{v_{ij}^k\}$ solve the dual to the kth subproblem, and any positive coefficient $(v_{ij}^k + v_{ji}^k)$, and the term $(v_{ij}^k + v_{ji}^k) y_{ij}$, of (1.10) can be interpreted as a potential reduction in the shortest path distance between nodes O(k) and D(k) caused by setting $y_{ij} = 1$ and introducing arc $\{i,j\} \varepsilon A$. The coefficient $(\Sigma_k v_{ij}^k + v_{ji}^k)$ of y_{ij} represents this total potential savings (generally an overestimate) over all commodities. The reduction might not be realized because arc $\{i,j\}$ need not lie on the shortest path joining some of the origins and destinations. Moreover, the savings of introducing two arcs $\{i,j\}$ and $\{r,s\}$ might not achieve the total potential $\Sigma_{k\varepsilon K}$ $(v_{ij}^k + v_{ji}^k) + \Sigma_{K\varepsilon K}$ $(v_{rs}^k + v_{sr}^k)$ because of interaction effects; that is, even if a new shortest path joining the origin and destination of some commodity k were to use both arcs {i,j} and {r,s}, we need not realize $v_{ij}^k + v_{ji}^k + v_{rs}^k + v_{sr}^k$ in savings. For that reason, the righthand side of (1.10) is a lower bound on the total cost z, and need not necessarily be equal to this cost.

2. STRENGTHENING BENDERS CUTS

The usual (or standard) Benders cut (1.10) is but one of many possible lower bounds on the cost of the network design problem. As we have already noted, this cut is derived from dual variables to the linear programming relaxation of the problem which is a shortest path problem. Since network problems are typically degenerate, their dual problems typically have alternate optimal solutions each defining a Benders cut. Are some of these cuts to be preferred to others? How might better cuts be generated?

In this section, we discuss five other Benders cuts. The first three are known lower bounds on the network design problem's objective value derived previously in the literature from arguments unrelated to Benders decomposition. We show that each of these lower bounds is a Benders cut derived by a judicious choice of the dual variables or the tentative network configuration used to generate the cut. This demonstration illustrates the richness of Benders cuts for the network design problem as well as the fundamental role of duality (in generating the Benders cut) for developing objective function bounds.

The fourth cut, which is new, dominates (in a way to be described later) the usual cut (1.10). Like some of these other lower bounds, it sharpens the cut inequality by considering penalties for eliminating arcs from the current configuration as well as savings attributed to adding new arcs (as in the usual cut). The fifth cut, which is of this same variety, has no explicit representation. Rather, it is generated by solving an auxiliary linear program to the subproblem which selects a "good" alternate solution to the dual. Although this procedure is a specialization of a more general cut generation procedure (Magnanti and Wong [1981]), because of the special structure of the network design problem it can be implemented efficiently by solving one network flow problem for generating each cut.

We might note that the general conception in this section of strengthening Benders cuts applies to other problem settings as well. In particular, similar results are possible in the context of facility location problems (Magnanti and Wong [1984b]), where the Benders cuts are related to the theory of submodular inequalities (Nemhauser and Wolsey [1981]).

2.1 Interpreting Lower Bounds as Benders Cuts

Previously, Boyce *et al.* [1973] proposed the following lower bound on the total cost z of the network design problem:

$$z \geq \sum_{\{i,j\}\varepsilon A} F_{ij} y_{ij} + R(y^a) \tag{2.1}$$

In this inequality, y^a denotes a candidate design that includes all possible arcs, i.e., $y^a_{ij} = 1$ for all $\{i,j\}\varepsilon A$. We shall refer to this special candidate network as the <u>all-inclusive design</u>. Recall that R(y) denotes the total (minimum cost) routing cost for the given candidate network y. Therefore, since all arcs are available in the all-inclusive design, $R(y^a)$ is the minimal possible routing cost for the problem. In particular, since $R(y) \geq R(y^a)$ for any network configuration y, (2.1) is a valid lower bound on total costs. Notice that the righthand side of this lower bound is actually a lower bounding function on total cost z(y) as a function of y. We have subsumed the dependence on y to conform with standard conventions from the literature on Benders cuts.

Independently, Gallo [1983] and Magnanti and Wong [1984] proposed another lower bound:

$$z \geq \sum_{\{i,j\}\varepsilon A} F_{ij} y_{ij} + \sum_{\{i,j\}\varepsilon A} y_{ij}^{c} [\sum_{k\varepsilon Q_{ij}} I_{ij}^{k}(y^{a})] + R(y^{a}) \qquad (2.2)$$

The formidable looking second term in this lower bound, which adds penalties for eliminating arcs from the all-inclusive design, requires some explanation. The variable $y_{ij}^{c} \equiv 1-y_{ij}$ equals 1 if arc $\{i,j\}$ is eliminated from the all-inclusive design and has value zero, otherwise. The term within brackets multiplying y_{ij}^{c} is a routing cost penalty for eliminating arc $\{i,j\}$ from the all-inclusive design. The sets Q_{ij} define any partition of the commodities and $I_{ij}^{k}(y^{a})$ is the incremental routing cost for commodity k incurred by deleting arc $\{i,j\}$ from the all-inclusive design. Note that this incremental cost term is zero if and only if some shortest path from O(k) to D(k) does not contain arc $\{i,j\}$.

Notice that since each y_{ij}^{c} and each incremental cost $I_{ij}^{k}(y^{a})$ is nonnegative, if any coefficient $I_{ij}^{k}(y^{a})$ is positive the lower bound (2.2) dominates the lower bound (2.1) in the sense that the righthand side of (2.2) is as large as the righthand side of (2.1) for all values of y and is strictly larger for at least one value of y.

We next show how to interpret both lower bounds (2.1) and (2.2) as Benders cuts. For (2.1), the interpretation is easy to derive. Simply consider the Benders cut generated by the all-inclusive design. Since every arc $\{i,j\}\varepsilon A$ belongs to this network configuration, for each commodity k, the shortest path distances u_i^k from O(k) to node i satisfy the optimality conditions $c_{ij}^k + u_i^k - u_j^k \geq 0$. Consequently, the v_{ij}^k and v_{ji}^k in (1.11) are zero and the usual cut (1.10) becomes the lower bound (2.1).

Showing that (2.2) is a Benders cut is a bit more difficult. To establish this fact, consider a given arc {g,h} and a given commodity $k \varepsilon Q_{gh}$. Suppose that we solve a shortest path problem for commodity k on the all-inclusive network with cost data specified as follows:

$$\tilde{c}_{ij}^k = c_{ij}^k \text{ and } \tilde{c}_{ji}^k = c_{ji} \quad \text{if } \{i,j\} \neq \{g,h\}$$

$$\tilde{c}_{gh}^k = c_{gh}^k + I_{gh}^k(y^a)$$

$$\tilde{c}_{hg}^k = c_{hg}^k + I_{gh}^k(y^a) \quad .$$

Let $\tilde{u}_i^k$ denote the shortest path distance from node O(k) to node i, with respect to these costs. For all $\{i,j\} \varepsilon A$, let $\tilde{v}_{ij}^k$ be defined by (1.11) with $u_i^k = \tilde{u}_i^k$;i.e., $\tilde{v}_{ij}^k = \max(0, \tilde{u}_j^k - \tilde{u}_i^k - c_{ij}^k)$ and $\tilde{v}_{ji}^k = \max(0, \tilde{u}_i^k - \tilde{u}_j^k - c_{ji}^k)$.

<u>Property 1:</u> The $\tilde{u}_i^k$ and $\tilde{v}_{ij}^k$ are feasible variables in the dual of the kth subproblem $S_k(y^a)$ corresponding to the all-inclusive design.

<u>Proof:</u> By substitution in the dual problem. (For any values of $\tilde{u}_i^k$, the $\tilde{v}_{ij}^k$ defined in this way correspond to a dual feasible solution.) □

<u>Property 2:</u> The $\tilde{u}_i^k$ and $\tilde{v}_{ij}^k$ are optimal variables in the dual of the kth subproblem $S_k(y^a)$.

<u>Proof:</u> Since these variables are dual-feasible, it suffices to show that their dual objective value equals the length of the shortest path from O(k) to D(k) in the all-inclusive design.

First, note that by definition of the incremental cost $I^k_{gh}(y^a)$, the length $\tilde{u}^k_{D(k)} - \tilde{u}^k_{O(k)}$ of the shortest path for commodity k in the all-inclusive network with costs $\tilde{c}^k_{ij}$ equals $I^k_{gh}(y^a)$ plus the shortest path length for costs c^k_{ij} . That is,

$$\tilde{u}^k_{D(k)} - \tilde{u}^k_{O(k)} = R_k(y^a) + I^k_{gh}(y^a). \qquad (2.3)$$

Next note that since $\tilde{u}^k_i$ are shortest path distances on the all-inclusive network, the optimality conditions $\tilde{c}^k_{ij} + \tilde{u}^k_i - \tilde{u}^k_j \geq 0$ are valid for all $\{i,j\} \varepsilon A$. But then since $c^k_{ij} = \tilde{c}^k_{ij}$ and $c^k_{ji} = \tilde{c}^k_{ji}$ for all $\{i,j\} \neq \{g,h\}$, $\tilde{v}^k_{ij} = 0$ and $\tilde{v}^k_{ji} = 0$ for all $\{i,j\} \neq \{g,h\}$.

Now consider arc $\{g,h\}$. If $I^k_{gh}(y^a) = 0$, then $\tilde{c}^k_{gh} = c^k_{gh}$, $\tilde{c}^k_{hg} = c^k_{hg}$ and, therefore, $\tilde{v}^k_{gh} = \tilde{v}^k_{hg} = 0$. If $I^k_{gh}(y^a) > 0$, then arc $\{g,h\}$ must lie in a shortest path P in the all-inclusive network with arc cost c^k_{ij}. Moreover, this path has length $R_k(y^a) + I^k_{gh}(y^a)$ in the all-inclusive network with the costs $\tilde{c}^k_{ij}$. Consequently, by (2.3) arc $\{g,h\}$ must also lie in a shortest path in this network and, thus, either $\tilde{c}^k_{gh} + \tilde{u}^k_g - \tilde{u}^k_h = 0$ or $\tilde{c}^k_{hg} + \tilde{u}^k_h - \tilde{u}^k_g = 0$. Assume the former (the argument is the same in either case.) Then $\tilde{v}^k_{gh} = \tilde{u}^k_g - \tilde{u}^k_h - c^k_{gh} = I^k_{gh}(y^a) > 0$. Finally, note that this conclusion implies that $\tilde{v}^k_{hg} = 0$, for otherwise summing $\tilde{v}^k_{gh}$ and $\tilde{v}^k_{hg} = \tilde{u}^k_h - \tilde{h}^k_g - c^k_{hg} > 0$ gives $- c^k_{gh} - c^k_{hg} > 0$, which is a contradiction. Note that whether $I^k_{gh}(y^a)$ equals zero or not, we may assume that $\tilde{v}^k_{gh} = I^k_{gh}$ and $\tilde{v}^k_{hg} = 0$

Using all of these computed values for the $\tilde{v}^k_{ij}$ and (2.3), now evaluate the dual objective function corresponding to $\tilde{u}^k_i$ and $\tilde{v}^k_{ij}$. Its value is given by

$$\tilde{u}^k_{D(k)} - \tilde{u}^k_{O(k)} - \sum_{\{i,j\}\varepsilon A} (\tilde{v}^k_{ij} + \tilde{v}^k_{ji})y^a_{ij} = [R_k(y^a) + I^k_{gh}(y^a)] - \tilde{v}^k_{gh}\, y^a_{gh}$$

$$= R_k(y^a) + I^k_{gh}(y^a) - I^k_{gh}(y^a) = R_k(y^a).$$

Since this value agrees with the routing cost for the kth subproblem, the $\tilde{u}^k_i$ and $\tilde{v}^k_{ij}$ are optimal dual variables. □

By Property 2, we can use $\{\tilde{u}^k_i, \tilde{v}^k_{ij}\}_{k\varepsilon K}$ to produce the Benders cut

$$z \geq \sum_{k\varepsilon K} [(\tilde{u}^k_{D(k)} - \tilde{u}^k_{O(k)}) - \sum_{\{i,j\}\varepsilon A} (\tilde{v}^k_{ij} + \tilde{v}^k_{ji})y_{ij}] + \sum_{\{i,j\}\varepsilon A} F_{ij}y_{ij}$$

or

$$z \geq \sum_{k\varepsilon K} R_k(y^a) + \sum_{\{i,j\}\varepsilon A} \sum_{k\varepsilon Q_{ij}} I^k_{ij}$$

$$- \sum_{\{i,j\}\varepsilon A} \sum_{k\varepsilon Q_{ij}} I^k_{ij}y_{ij} + \sum_{\{i,j\}\varepsilon A} F_{ij}y_{ij}$$

which, upon substituting $y^c_{ij} = 1-y_{ij}$, is exactly (2.2). So the lower bound inequality proposed by Gallo and Magnanti and Wong can be viewed as a Benders cut.

Hoang [1973] has proposed a cut that is an intermediary between (2.1) and (2.2); it is obtained by setting $Q_{O(k),D(k)} = \{k\}$ for all $k\varepsilon K$. Therefore, the set Q_{ij} partitions the commodities into singletons and $I^k_{ij}(y^a)$ for $i = O(k)$ and $j = D(k)$ corresponds to the incremental cost between the origin and destination of commodity k if we eliminate the possibility of single-arc direct routing. Here, we have assumed that $O(k_1) = D(k_2)$ and $D(k_1) = O(k_2)$ is never true; otherwise, we would set $Q_{O(k_1),\, D(k_2)} = \{k_1,k_2\}$ and the interpretation would be similar.

Figure 2.1 illustrates an example of the usual and improved Benders cuts we have discussed. In this example, one unit of flow is to be sent from node 1 to node 3 and another unit from node 1 to node 4. The set A consists of all arcs drawn in the figure. The arc labels specify the arc routing costs which are the same for both commodities. Let k=3 and k=4 refer to the two destination nodes of the commodities. The optimal dual variable values for u_i^k with k=3 or 4 have three different values corresponding to the usual, Hoang, and Gallo-Magnanti-Wong Benders cuts.

The first set of dual variables yields Boyce _et al._'s usual cut, corresponding to (2.1)

$$z \geq \sum_{\{i,j\}\varepsilon A} F_{ij}y_{ij} + [(2+5) = 7].$$

The second set of dual variables indicates that deleting arc {1,3} from the network increases the routing cost between nodes 1 and 3 from $u_1^3 + 2 = 2$ to $u_2^3 = 11$ for a net increase of 9 units (i.e., $I_{13}^3 = 9$). Thus, Hoang's cut is

$$z \geq \sum_{\{i,j\}\varepsilon A} F_{ij}y_{ij} + 7 + 9(1 - y_{13}).$$

The third set of dual variables corresponds to the Gallo-Magnanti-Wong cut (2.2) where commodity 3 belongs to Q_{13} and commodity 4 belongs to Q_{34}. This cut provides the additional information that deleting arc {3,4} from the network increases the routing cost between nodes 1 and 4 from $u_1^4 + 5 = 5$ to $u_4^4 = 9$ for a net increase of 4 units (i.e., $I_{34}^4 = 4$ and $I_{13}^3 = 9$). The resulting cut is

$$z \geq \sum_{\{i,j\}\varepsilon A} F_{ij}y_{ij} + 7 + 9(1-y_{13}) + 4(1-y_{34}) \ .$$

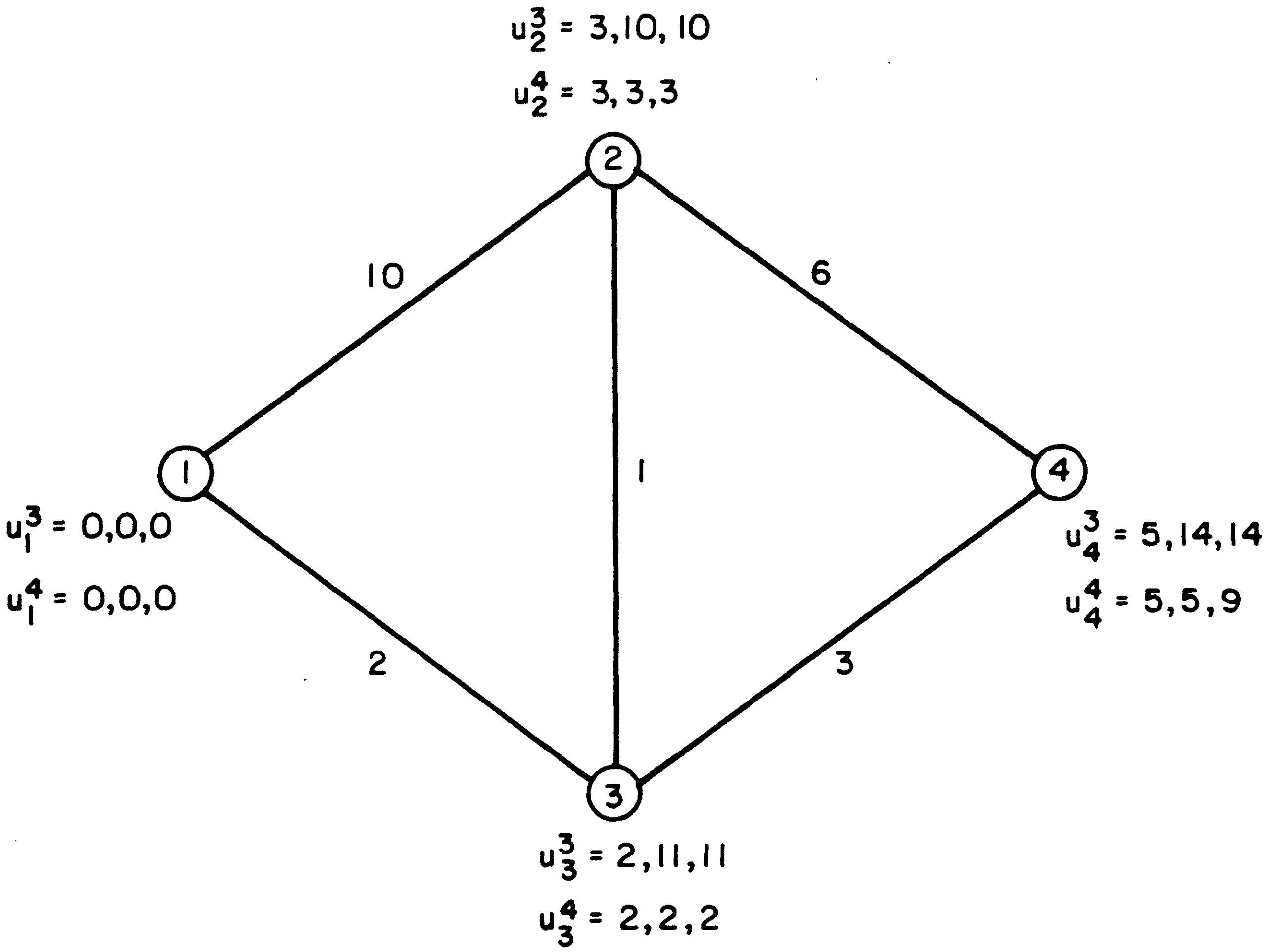

Figure 2.1 A Step of Benders Decomposition with Alternate Dual Solutions

This example illustrates the potential for strengthening the usual Benders cut by incorporating information about the penalties for deleting arcs in the current design. Although we discussed these ideas for only the specific design $y = y^a$, they can easily be adapted to any arbitrary candidate network.

2.2 Strong Benders Cuts for the Network Design Problem

In Section 1, we saw that the optimal dual variables for the usual Benders cut could be interpreted as upper bounds for estimating the decrease in network routing cost due to the addition of arcs not in the current design $\bar{y}$. The previous subsection demonstrated how the usual Benders cut could be improved by considering penalties caused by removing an arc already in the current design.

Next, we discuss another technique for strengthening the usual Benders cuts which utilizes more accurate estimates of the decrease in routing cost due to the addition of an arc. As before, we introduce this new Benders cut as an alternate optimal dual solution to the Benders subproblem.

Let $\{\hat{u}_i^k, \hat{v}_{ij}^k\}_{k\epsilon K}$ be the dual variable values corresponding to the usual Benders cut for a candidate design $y = \bar{y}$. We define a "strong" Benders cut by specifying a new dual solution $\{\bar{u}_i^k, \bar{v}_{ij}^k\}_{k\epsilon K}$ as follows:

$$\bar{u}_{D(k)}^k = \hat{u}_{D(k)}^k = \text{routing cost for commodity } k$$

$$\bar{u}_{O(k)}^k = \hat{u}_{O(k)}^k = 0$$

$$\Delta_i^k \equiv \bar{u}_{D(k)}^k - D_i^k$$

$$\bar{u}_i^k = \min(\hat{u}_i^k, \Delta_i^k) \quad i \neq O(k) \text{ or } D(k)$$

$$\bar{v}_{ij}^k = \max\{0, \bar{u}_j^k - \bar{u}_i^k - c_{ij}\} \tag{2.4}$$

In the expression defining Δ_i^k, D_i^k denotes the minimum cost of routing one unit of commodity k from node i to node D(k) on the all-inclusive network given by $y = y^a$.

As noted in Section 1, for the usual Benders cut with the dual variables $\{\hat{u}_i^k, \hat{v}_{ij}^k\}_{k\varepsilon K}$, we could interpret

$$\hat{v}_{ij}^k = \max(0, \hat{u}_j^k - \hat{u}_i^k - c_{ij}^k)$$

as an upper bound on the decrease in the routing cost for commodity k if arc {i,j} is added to the current design $y = \bar{y}$.

Although the strong cut dual variables have a similar interpretation, they provide lower bound estimates that are no worse than, and usually improve upon, the ones provided by the usual cut. That is, every $\bar{v}_{ij}^k \leq \hat{v}_{ij}^k$.

<u>Property 3:</u> a) For every $k\varepsilon K$, $\{\bar{u}_i^k, \bar{v}_{ij}^k\}$ is an optimal solution for the dual to the kth subproblem $S_k(\bar{y})$.

b) $\bar{v}_{ij}^k \leq \hat{v}_{ij}^k$ and $\bar{v}_{ji}^k \leq \hat{v}_{ji}^k$ for all $\{i,j\}\varepsilon A$ and all $k\varepsilon K$.

<u>Proof:</u> We begin by proving $\bar{v}_{ij}^k \leq \hat{v}_{ij}^k$ in part b.

<u>Case 1:</u> Suppose $\bar{u}_i^k = \hat{u}_i^k$. Then since $\bar{u}_j^k \leq \hat{u}_j^k$, we have $\bar{u}_j^k - \bar{u}_i^k \leq \hat{u}_j^k - \hat{u}_i^k$ which by the definitions (1.11) and (2.4) of the v_{ij}'s implies that $\bar{v}_{ij}^k \leq \hat{v}_j^k$.

<u>Case 2:</u> Suppose $\bar{u}_i^k = \Delta_i^k$. The triangle inequality implies that

$$D_i^k \leq D_j^k + c_{ij}^k$$

or

$$D_i^k - c_{ij}^k \leq D_j^k .$$

Consequently,

$$\Delta_j^k = \bar{u}_{D(k)}^k - D_j^k \leq \bar{u}_{D(k)}^k - (D_i^k - c_{ij}^k) = (\bar{u}_{D(k)}^k - D_i^k) + c_{ij}^k$$

and thus

$$\Delta_j^k \leq \Delta_i^k + c_{ij}^k \quad .$$

This inequality and the fact that $\bar{u}_j^k \leq \Delta_j^k$ implies that

$$\bar{u}_j^k - \bar{u}_i^k \leq \Delta_j^k - \bar{u}_i^k = \Delta_j^k - \Delta_i^k \leq \Delta_i^k + c_{ij}^k - \Delta_i^k$$

$$\text{or} \quad \bar{u}_j^k - \bar{u}_i^k \leq c_{ij}^k \quad .$$

From (2.4), this last inequality implies that $\bar{v}_{ij}^k = 0$, and since $\hat{v}_{ij}^k \geq 0$, we have $\bar{v}_{ij}^k \leq \hat{v}_{ij}^k$. A similar argument shows that $\bar{v}_{ji}^k \leq \hat{v}_{ji}^k$ and, thus, completesthe proof of part b.

In part a, substituting the values $\{\bar{u}_i^k, \bar{v}_{ij}^k\}$ into the dual of P_1 shows they are dual-feasible. By definition, $(\bar{u}_{D(k)}^k - \bar{u}_{O(k)}^k) = (\hat{u}_{D(k)}^k - 0)$ which is the optimal objective value to the dual of $S_k(\bar{y})$. Now $\bar{y}_{ij} = 1$ implies that $\bar{v}_{ij}^k = 0$ since $\bar{v}_{ij}^k \leq \hat{v}_{ij}^k$ by part (b) and $\hat{v}_{ij}^k = 0$ by the optimality condition $c_{ij}^k + \hat{u}_i^k - \hat{u}_j^k \geq 0$ of the shortest path subproblem $S_k(\bar{y})$. A similar argument shows that $\bar{y}_{ij} = 1$ also implies that $\bar{v}_{ji}^k = 0$. Therefore, the objective function value of the dual solution $\{\bar{u}_i^k, \bar{v}_{ij}^k\}$ is $\hat{u}_{D(u)}^k$ and so the solution is optimal as well as feasible.

Substituting $\{\bar{u}_i^k, \bar{v}_{ij}^k\}_{k \epsilon K}$ into (1.10) yields

$$z \geq \sum_{k \epsilon K} (\bar{u}_{D(k)}^k - \bar{u}_{O(k)}^k) - \sum_{\{i,j\} \epsilon A} (\bar{v}_{ij}^k + \bar{v}_{ji}^k) y_{ij} + \sum_{\{i,j\} \epsilon A} F_{ij} y_{ij}$$

or, since $\bar{u}_{D(k)}^k = \hat{u}_{D(k)}^k$ and $\bar{u}_{O(k)}^k = \hat{u}_{O(k)}^k = 0$,

$$z \geq \sum_{k \epsilon K} (\hat{u}_{D(k)}^k - \hat{u}_{O(k)}^k) - \sum_{\{i,j\} \epsilon A} (\bar{v}_{ij}^k + \bar{v}_{ji}^k) y_{ij} + \sum_{\{i,j\} \epsilon A} F_{ij} y_{ij} \quad .$$

Since $\bar{v}_{ij}^k \leq \hat{v}_{ij}^k$ and $\bar{v}_{ji}^k \leq \hat{v}_{ji}^k$ for all $\{i,j\}$, the strong cut will be no worse than the usual cut and will frequently dominate it.

Figure 2.2 depicts an example comparing the two types of cuts. The design problem is the same as the one in Figure 2.1 except that the only flow requirement is to route one unit between nodes 1 and 4. Assume that the current design consists of all arcs except for arc $\{2,3\}$.

The first set of dual variables corresponds to the usual cut

$$z \geq 5 - 7y_{23}.$$

The second set of dual variables leads to the strong cut

$$z \geq 5.$$

The usual cuts estimate that arc $\{2,3\}$ would belong to a shortest path between nodes 1 and 4 and potentially save $u_2^4 - u_3^4 - c_2^3 = 10-2-1 = 7$ units of cost. Looking ahead, though, from node 3 to 4 by computing the shortest path distance from every node to node 4, we "see" the high cost of arc $\{2,4\}$. Thus, by looking for additional costs "further down the road," we are able to produce a more accurate estimate of the value of adding arc $\{2,3\}$ to the network design.

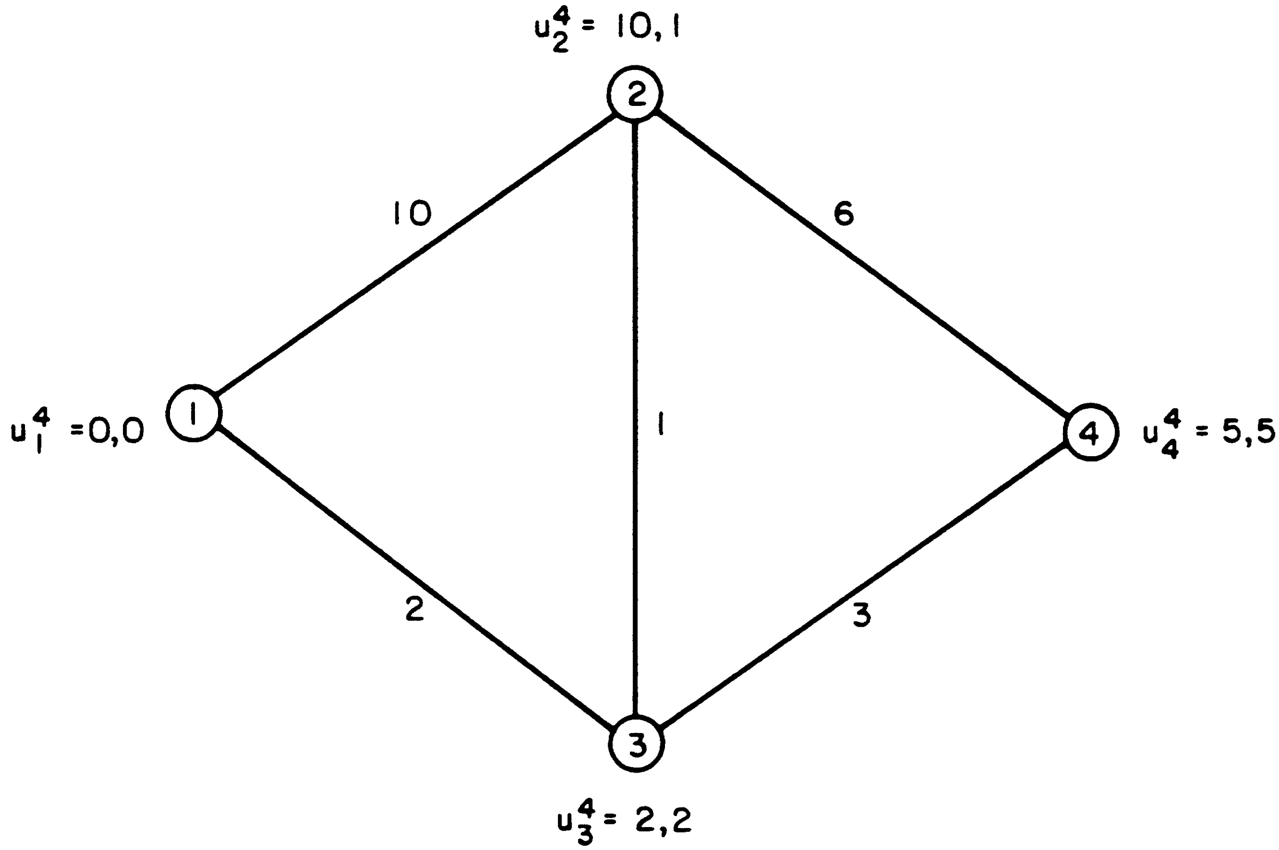

Figure 2.2 One Step of Benders Decomposition with Usual and Strong Cuts

2.3 Pareto-Optimal Cuts for Network Design Problems

To this point, we have seen several methods for generating Benders cuts for the network design problem that "improve" upon the usual ones in the sense that they are no worse than, and usually dominate, the usual cuts. In general, we would prefer to have cuts that are not dominated by any other cut. We call such undominated cuts Pareto-optimal.

Magnanti and Wong [1981] have introduced a general methodology for generating Pareto-optimal Benders cuts for any mixed integer programming problem. Their method conveniently specializes to the network design problem in the following way. Suppose that we have solved the dual to the subproblem $S_k(\bar{y})$ for the current design $y = \bar{y}$. Now consider the auxiliary linear program:

$$(AP_k(\bar{y})): \qquad \text{maximize } (u^k_{D(k)} - u^k_{O(k)}) - \sum_{\{i,j\}\varepsilon A} (v^k_{ij} + v^k_{ji})y^0_{ij}$$

$$u^k_{D(k)} - u^k_{O(k)} - \sum_{\{i,j\}\varepsilon A} (v^k_{ij} + v^k_{ji})\bar{y}_{ij} \geq (R_k\ \bar{y})$$

$$\begin{aligned} &u^k_j - u^k_i - v^k_{ij} \leq c^k_{ij} && \text{all } \{i,j\}\varepsilon A \\ &u^k_i - u^k_j - v^k_{ji} \leq c^k_{ji} && \text{all } k\varepsilon K \\ &v^k_{ij}\ ,\ v^k_{ji} > 0 \end{aligned}$$

where y^0 is a point in the relative interior of the convex hull of Y. Notice that $AP_k(\bar{y})$ is similar to the dual $S_k(\bar{y})$ except that it has a different objective function and an additional constraint that restricts the feasible region to the set of optimal solutions to this dual problem.

Suppose that we find an optimal solution for $AP_k(\bar{y})$. Magnanti and Wong proved that the corresponding cut is Pareto-optimal.

For the network design problem, Benders decomposition can produce Pareto-optimal cuts at the price of solving an additional |K| auxiliary linear programs. In fact, it can do so by solving |K| minimum cost flow problems. To demonstrate this fact, we form the linear programming dual of $AP_k(y)$.

$$\text{Min} \sum_{\{i,j\}\varepsilon A} (c^k_{ij}x_{ij} + c^k_{ji}x_{ji}) - R_k(\bar{y})x_0$$

subject to:

$$\sum_j x_{js} - \sum_j x_{sj} = -(1 + x_0) \qquad \text{where } s = O(k)$$

$$\sum_j x_{ji} - \sum_j x_{ij} = 0 \qquad \text{for all } i \neq O(k) \text{ or } D(k) \qquad (2.5)$$

$$\sum_j x_{jt} - \sum_j x_{tj} = +(1 + x_0) \qquad \text{where } t = D(k)$$

$$\left.\begin{array}{l} x_{ij} \leq y^0_{ij} + x_0\bar{y}_{ij} \\ x_{ji} \leq y^0_{ij} + x_0\bar{y}_{ij} \end{array}\right\} \qquad \text{for all } \{i,j\}\varepsilon A$$

$$x_{ij} \geq 0, \quad x_0 \geq 0 .$$

In this formulation x_0 is the dual variable associated with the first constraint of problem $AP_k(\bar{y})$ and x_{ij} are dual variables associated with the other constraints. This reformulation shows that we are dealing with a parametric minimum cost flow problem in the scalar parameter x_0 which induces a variable demand of $(1 + x_0)$ units, a variable arc capacity of $y^0_{ij} + x_0$ for the arcs in the current design (i.e., with $\bar{y}_{ij} = 1$), and a fixed arc capacity of y^0_{ij} for the arcs not in the current design (i.e., with $\bar{y}_{ij} = 0$). We receive a rebate of $R_k(\bar{y})$ for each unit after the first one that is sent from O(k) to D(k) and must send a flow of at least one unit from O(k) to D(k). Once that unit is sent, if the marginal cost of sending additional units is less than $R_k(\bar{y})$, there is incentive to increase the value of x_0 until the marginal cost becomes greater than or equal to $R_k(\bar{y})$. At this point, no further increase in the value of x_0 can improve the objective function of (2.5)

We claim that any value of $x_0 \geq \sum_{\{i,j\}\varepsilon A} y_{ij}^0$ is optimal in (2.5). To establish this fact, note that the total capacity $y_{ij}^0 + x_0\bar{y}_{ij}$ for arcs not in the current design (those with $\bar{y}_{ij} = 0$) is at most $\sum_{\{i,j\}\varepsilon A} y_{ij}^0$. Also, each unit of the $(1+x_0)$ demand must flow along some path(s) from O(k) to D(k). At most $\sum_{\{i,j\}\varepsilon A} y_{ij}^0$ of these units can use path(s) containing an arc not in the current design defined by $\bar{y}$ since each unit of flow along such paths must use at least one unit of capacity of arcs not in the current design. Any additional flow must use path(s) that contain arcs only in the current design and, therefore, their marginal cost will be $R_k(\bar{y})$.

So any value of $x_0 \geq \sum_{\{i,j\}\varepsilon A} y_{ij}^0$ must be optimal for (2.5).

By fixing $x_0 \geq \sum_{\{i,j\}\varepsilon A} y_{ij}^0$, we can solve (2.5) as a minimum cost flow problem and the optimal dual variables will be an optimal solution for $AP_k(\bar{y})$. Solving $|K|$ such minimum cost flow problems determines the coefficients $\{u_i^k, v_{ij}^k\}_{k\varepsilon K}$ for a Pareto-optimal Benders cut.

Note that like all of the procedures given in this section for generating improved cuts, the Pareto-optimal cut algorithm requires more computational effort than the usual cut algorithm. However, the improved cuts should accelerate the convergence of the master problem and thus decrease the overall computation time of Benders decomposition. Our computational results in Section 4 confirm this suspicion.

3. PROBLEM PREPROCESSING

In the past several years, researchers have made substantial progress in solving integer programs. Indeed, recent computational experience has demonstrated an important lesson--the synthesis of varied solution strategies often significantly extends solution capabilities and permits integer programming algorithms to solve large-scale applications. Moreover, the research community is witnessing a resurgence of several previously discarded methods such as cutting planes. This renaissance and the improved performance of these methods is attributable in part to the way that these methods are now being integrated with other solution approaches.

For example, Crowder and Padberg [1980], Crowder, Johnson, and Padberg [1983], Barany, Van Roy, and Wolsey [1983], and Martin and Schrage [1982] all have successfully solved large-scale integer programming problems by using cutting planes of the integer programming feasible region. These methods incorporate facet generating inequalities, logical inequalities, coefficient reduction, and/or variable elimination procedures within a branch and bound framework. Lemke and Spielberg [1967], Guignard and Spielberg [1981], and Guignard [1982], have made similar proposals, and Geoffrion and Marsten [1972] and Land and Powell [1979, 1981] have given integrating frameworks of integer programming methods and surveys of computational codes and computational experience.

For specific applications to the network design problem, Billheimer and Gray [1973] used methods for eliminating variables with a heuristic local improvement scheme.

In preliminary testing with pure Benders decomposition for the network design problem, we observed that the algorithm's performance was quite erratic for larger-scale problems with more than 50 integer variables. This observation prompted us to consider preprocessing procedures that would

(i) eliminate variables to reduce problem size;

(ii) incorporate information from the linear programming relaxation for bounding purposes; and

(iii) produce logical constraints that further restrict the integer variables.

Our preprocessing procedure works by first obtaining a feasible solution to the dual of the linear programming relaxation of the network design formulation (1.1)-(1.5). (In fact, the routine uses a slightly more complicated formulation of the network design problem adopted from Magnanti and Wong [1984c]. We will not discuss this modified integer programming formulation since it would unnecessarily encumber the presentation of our main ideas.)

Notice that any dual feasible solution (u,v) to the full linear programming relaxation is also a feasible solution to the dual for the Benders subproblem. Rewriting (1.10) using a dual feasible solution produces the Benders cut

$$z \geq a_0 + \sum_{\{i,j\}\epsilon A} a_{ij}y_{ij} \tag{3.1}$$

where

$$a_0 = \sum_{k\epsilon K} (u^k_{D(k)} - u^k_{O(k)})$$

and

$$a_{ij} = F_{ij} - \sum_{k\epsilon K} (v^k_{ij} + v^k_{ji}) \quad \text{for all } \{i,j\}\epsilon A \ .$$

Inspection of (1.1) -(1.5) and the dual feasibility of (u,v) allows us to interpret a_0 as the dual objective function value of (u,v) and each a_{ij} as the nonnegative slack in the dual constraint corresponding to the variable y_{ij}.

We use a heuristic technique known as dual ascent to generate a dual feasible solution. The procedure exploits the simple form of the objective function and iteratively increases each $u^k_{D(k)}$ variable and adjusts the other variables in order to preserve feasibility. The algorithm terminates when it reaches a local optimum and no further change of a single $u^k_{D(k)}$ variable can improve the objective function. Note that by linear programming duality theory the dual objective function value a_0 is a lower bound for the value of the linear programming relaxation and consequently, for the value of the integer programming problem (1.1)-(1.5). We can also use the dual ascent solution to derive an upper bound a_1 for the optimal design cost. To do so, we form a candidate design consisting of all arcs {i,j} whose dual slack a_{ij} is zero in the final solution. Then we improve this design by applying a simple drop-add local improvement heuristic to generate a feasible solution to the problem. The full details of this dual ascent-based algorithm for obtaining upper and lower bounds for the network design problem are rather complicated and will be given in a separate paper (Magnanti and Wong [1984c]).

A number of authors have successfully used dual ascent procedures to solve combinatorial optimization algorithms. The method has produced excellent computational results in solving large-scale problems for uncapacitated plant location (Bilde and Krarup [1977] and Erlenkotter [1978]), for plant location with side constraints (Guignard and Spielberg [1979]), for data base location (Fisher and Hochbaum [1980]), for generalized assignment (Fisher, Jaikumar and Van Wassenhove [1980], for dynamic plant location (Van Roy and Erlenkotter [1982]), for asymmetric traveling salesman problems (Balas and Christofides [1981]), and for the Steiner tree problem on a graph (Wong [1984]). In keeping with the objectives of this study, we decided not to implement dual ascent as a solution technique by itself, but rather to use it in a variable elimination routine for preprocessing.

The elimination routine works in the following way. We use the dual ascent-based algorithm to obtain upper and lower bounds a_1 and a_0 on the optimal objective value to the design problem as well as a Benders cut (3.1). Consider any arc $\{g,h\}$ with $a_{gh} > (a_1-a_0)$. Then arc $\{g,h\}$ can be eliminated from the problem since no design with arc $\{g,h\}$ can be optimal. To see this, recall that the coefficients a_{ij} in the dual ascent-based Benders cut (3.1) are nonnegative. Therefore,

$$a_0 + a_{gh} > a_1$$

is a lower bound for the cost of any design containing arc $\{g,h\}$ and any design with arc $\{g,h\}$ is worse than the design corresponding to the upper bound a_1.

The preprocessing routine also derives other information from the dual ascent solution. Let $S = [\{i,j\}\varepsilon A : a_{ij} > (a_1 - a_0)/2]$. Any optimal design solution must satisfy the multiple choice constraint

$$\sum_{\{i,j\}\varepsilon S} y_{ij} \leq 1 \quad . \tag{3.2}$$

Any solution containing two or more members $\{i_1,j_1\}$ and $\{i_2,j_2\}$ of S cannot be optimal since it has a cost of at least

$$a_0 + a_{i_1j_1} + a_{i_2j_2} > a_0 + (a_1-a_0)/2 + (a_1-a_0)/2 = a_1$$

Although several other inequalities like (3.2) are possible, we have limited our implementation to this multiple choice constraint.

The overall preprocessing routine works by applying the dual ascent-based algorithm to eliminate as many variables as possible. We then reapply the dual ascent routine to the reduced network design problem and try to eliminate additional variables. This process continues until no further variable elimination is possible and a minimum of 5 dual ascent iterations have been performed. The final dual solution also generates a Benders cut (3.1) and a logical constraint (3.2). Notice how important it is for the ascent-based procedure to produce close upper and lower bounds in order for the variable elimination routine and the generated constraints (3.1) and (3.2) to be effective.

4. COMPUTATIONAL RESULTS

Our computational results tested three versions of Benders decomposition (with the usual cuts, strong cuts, and Pareto-optimal cuts) as well as the preprocessing procedures discussed in the last section. We undertook two sets of experiments:

(i) using a pure Benders decomposition on a set of test problems with up to 45 0-1 variables; and

(ii) using Benders decomposition with preprocessing on a set of test problems with up to 90 0-1 variables.

In each case, we implemented all algorithms in FORTRAN on a VAX 11/780 or a CYBER 76 computer. We solved master problems using a rudimentary linear programming- based branch and bound code (Land and Powell [1973]). The implementation used a state-of-the-art primal simplex code (Kennington and Helgason [1980]) to solve the minimum cost network flow problems required for generating Pareto-optimal cuts. We also used a naive implementation of Dijkstra's algorithm for finding shortest paths. Since most of the computation time is spent solving the master problem, a more efficient shortest path algorithm would not have produced any significant improvement in the overall computation time.

4.1 Benders Decomposition Without Problem Preprocessing

The first sets of experiments clearly demonstrated the superiority of both the strong and Pareto-optimal cuts over the usual cuts, though as the problem size grew, the increase in computational time led us to consider problem preprocessing.

Tables 1 and 2 specify the results for the first set of test problems. For these experiments, we used a problem generation procedure that selected nodes randomly from a 50x100 rectangle in the plane. The procedure generated the arc set A by randomly choosing arcs from the set $\{\{i,j\}: i \in N, j \in N\}$ so that each was equally likely to be chosen, using an efficient procedure discussed in Knuth [1969, section 3.4.2]. In addition, for some of the tests we discarded arcs whose distance exceeded a maximum distance problem parameter. Although this procedure does not guarantee feasibility, all of the test problems generated were feasible.

Arc Costs and Commodity Types

We selected the routing cost c_{ij} for each arc $\{i,j\}$ in several ways. In some cases, we chose the routing costs randomly using a uniform random variable U(a,b) defined on the interval [a,b]. In other cases, we let the cost on an arc be equal to its Euclidean length, independent of commodity type. The fixed cost F_{ij} for arc $\{i,j\}$ was either a constant times c_{ij} or a constant minus c_{ij}. Thus, they were either proportional or inversely related to the routing costs. The set of commodities K consisted either of all origin-destination pairs or of all origin-destination pairs originating from two arbitrarily selected nodes.

Arcs That Are In Every Network Design

For all of the test problems, we modified the fixed costs in order to ensure that some arcs remain in all candidate designs. For these types of problems, which can be viewed as network improvement problems, Benders decomposition encounters fewer infeasible candidate solutions, and thus concentrates on cost trade-offs. The procedure specifies a set X of arcs from A by random selection, with the number #DARCS of these arcs fixed as a problem parameter. It then redefines the fixed costs of the remaining (#ARC - #DARCS) arcs at value zero. Since all such arcs can trivially be included in an optimal design, the number of actual 0-1 variables becomes #DARCS.

Tables of Computational Results

In tables 1 and 2 the entry for each test problem is grouped together into a set of rows headed by the problem name DATAXX. The first row with the entry DATAXX describes the number of nodes, arcs and commodities and the optimal solution value for the problem. The succeeding rows summarize computational experience on the application of Benders decomposition with a particular cut type.

The column headed #ITER specifies the number of times the master problem was solved for any problem. Of these, in #ITER1 iterations, the master problem was modified by the addition of feasibility cuts. The columns headed #CUTS and #FCUTS give the total number of cuts (objective function plus feasibility) used to solve the problem. Notice that #FCUTS exceeds #ITER1 since some iterations added more than one feasibility cut to the master problem (since the network had more than one O-D pair that was disconnected). The terms CPU and MAS% describe the total CPU time and the percentage of CPU time spent on solving the master problems. For those cases when Benders algorithm did not identify an optimal solution, UB and LB give the best upper and lower bounds obtained and %DIF = [(UB-LB)/UB] · 100% gives the percentage difference between the two bounds. For problems that terminated prematurely without a verified optimal solution, the letter after the CPU time indicates whether termination was due to a computer time limit (T), or due to a numerical instability and/or a choice of program parameters (e.g., maximum number of reinversions) in the Land and Powell integer programming routine that led to looping (L) in the algorithm or caused the system to terminate abnormally (A).

Looping occurs when the master problem continues to generate the same nonoptimal solution (i.e., the new cut does not eliminate the nonoptimal solution that generated it). The system terminated abnormally when it generated a lower bound on the objective function that exceeded the upper bound.

Every test problem has a COST ASSUMPTION entry that summarizes the way in which we generated the variable and fixed costs for the arcs.

Notice from Tables 1 and 2 that the problem sizes for these test problems ranged from #DARCS = 35 to #DARCS = 45 binary variables and from 2 (#ARCS) • $|K|$ = 810 to 105,600 continuous variables. (The factor of 2 is included in the last count because there are two flows x_{ij}^k and x_{ji}^k for each commodity k on each undirected arc $\{i,j\}$.)

Interpreting The Computational Results

Notice from the column entitled CPU in Tables 1 and 2 that the usual cut could solve only 3 of the 16 test problems in this first group within our time limit of about 2 hours. In fact, at termination the percent difference (%DIF) between the best lower bound and best feasible solution generated by the usual cuts was as much as 20% for over a quarter of the problems. On the other hand, the strong cuts solved all these problems within the time limit (averaging 963 seconds on problems DATA01 - DATA10, and 1264 seconds on problems DATA11 - DATA13A) and the Pareto-optimal cuts solved all but one of these test problems within this time limit (averaging 128 seconds on problems DATA01 - DATA10, and 1511 seconds on problems DATA11 - DATA13A). For most of the test problems, the Pareto-cut implementation required fewer cuts (objective function cuts #CUTS plus feasibility cuts #FCUTS) than the strong cut implementation, though at the expense of larger computation time for generating each cut. In none of the cases did the usual cut outperform the other two cut strategies in solution time or number of cuts. Finally, note from the column entitled MAS% that solving the master problem consumed about 100%, 88%, and 54% of the solution time of the usual, strong, and Pareto cuts in the test problems DATA01 - DATA10.

4.2 Benders Decomposition with Problem Preprocessing

For our second set of computational experiments, we tested Benders decomposition with our problem preprocessing algorithm on the previous set of 16 problems (DATA1 to DATA13A) and on a second set of test problems that contained 90 binary variables, except for three cases that had 45 binary variables. The number 2(#ARCS) $|K|$ of continuous variables was 4(#ARCS)(#NODES-1) since the problems had 2 source nodes each with a commodity destined for every other node. For these test problems, the number of continuous variables ranged from 6,700 to 15,080.

For the second set of test problems, we generated the node set, arc set, and set of arcs with zero fixed costs as before. We generated arc routing costs and fixed costs in several different ways.

Arc Costs

In defining arc costs, we divided the second set of test problems into several groups. For the first group of these problems (DATA14 to DATA18), we generated the routing and fixed costs for each arc by sampling from a uniform U(50,150) distribution and a uniform U(500,1000) distribution, respectively. Problems 19 and 20 also used uniform distributions for both the routing and fixed costs. The third group of test problems (DATA21 to DATA26) used arc routing costs drawn from a U(100,200) distribution and fixed costs equal to fixed constant times the arc routing cost. For the fourth group (DATA28 to DATA31), we let

$$\text{arc routing cost} = 10\ (\text{arc length}) + b_1$$

and

$$\text{arc fixed cost} = 2(\text{routing cost}) + b_2$$

where b_1 and b_2 are values drawn from a U(50,150) and a U(-100,100) distribution, respectively.

The remaining problems were generated by procedures that are variants of those used for these four groups of problems.

Tables of Computational Results

Tables 3 and 4 document the computational results of Benders algorithm for our two sets of test problems. These tables have the same layout as Tables 1 and 2 except for the additional results on the dual-ascent-based preprocessing routine. Tables 3 and 4 specify the preprocessing computation time, CPU, the number #RARCS of integer variables remaining after application of the preprocessing, and #MCHU the number of variables present in the logical multiple choice constraint (3.2). The columns headed UB and LB specify the best upper and lower bounds produced by the dual ascent-based algorithm. The column %DIF indicates the percentage difference [(UB-LB)/UB · 100%] between these two bounds. Notice that when UB = LB for the dual ascent routine, it found and verified an optimal solution. For these cases, the use of Benders decomposition was not necessary. When UB ≠ LB from the dual ascent, we applied Benders decomposition using the usual strong or pareto-optimal cuts.

Interpreting the Computational Results

Table 4 shows that Benders decomposition with preprocessing was able to solve to optimality 19 out of the last group of larger 24 test problems and to find solutions to 23 of the problems that are guaranteed to be within at most 1.44% of optimality. For one particularly difficult problem, the Benders algorithm did not improve upon the bounds generated by the dual ascent procedure, which found a solution guaranteed to be within 5.53% of optimality.

For seven of the problems (15,17,18,20,28,32, and 34), the preprocessing routine itself found and verified an optimal solution. For the remaining problems, the processing routine was usually able to eliminate 75 to 95% of a problem's zero-one variables.

For the 17 of these problems that used Benders decomposition, the strong cut version clearly dominated the one with usual cuts. For the 12 problems when the strong Benders cut implementation converged within our time limit, it was as fast as the usual cut implementation in one case, at least twice as fast in every other case, and as much as 145 times as fast (even without counting the additional time that would be required for the usual cut implementation to converge.) For the five problems where Benders algorithm using strong cuts terminated permaturely due to time limitations, the strong cuts showed a moderate advantage.

The Pareto-optimal cut version was usually more effective for the more difficult problems. For the five problems that terminated prematurely, the Pareto-optimal cuts performed better than either the strong or usual cuts. For the other 13 problems, the strong cuts out-performed the Pareto-optimal cuts, but the Pareto-optimal cuts improved considerably upon the usual cuts. Recall that generating Pareto-optimal cuts requires the solution of $|K|$ minimum cost network flow problems while generating strong cuts requires the solution to only $|0|$ shortest path tree problems (where $0 = \{i | i=0(k), k=1...K\}$ is the set of nodes that are the origin of at least one commodity). Therefore, the Pareto-optimal cut strategy might be expected to require more solution time than the strong cut strategy. On the other hand, the Pareto cuts seem to generate somewhat better cuts and this improvement seems to be more pronounced for the more difficult problems.

We might also note that the preprocessing routine procedures are effective in reducing problem sizes considerably. Indeed, the resulting reduced problems are as small as problems that could be solved by more straightforward branch and bound procedures. However, the reduced problems appear to be much more difficult than non-preprocessed problems of comparable size (e.g., the problems in our first set of test problems).

Comparing Tables 1 and 2 with Table 4 highlights this fact. Table 3 also shows that the dual ascent procedure is very effective in solving smaller-sized problems even without using preprocessing to eliminate variables. The overall performance of the algorithm is very encouraging, especially since the test problems were generated with such a wide range of cost data relationships. These results indicate that Benders decomposition with problem preprocessing could be an effective method for solving a broad class of large-scale network design problems.

Finally, we note, as reported in Table 4, that solving the master problem consumed a large proportion of the solution time for every implementation of Benders decomposition that we tested. In particular, it consumed over 90% of the computation time for the five most difficult problems--numbers 29, 30, 35, 36, and 37 (an appendix gives the data for these problems). Consequently, solving the master integer programming problem with a commercial branch and bound code or by a specialized implicit enumeration algorithm could conceivably have dramatic effects upon the algorithm's performance, and might enable the procedures described in this paper to solve very large-scale applications.

Acknowledgement: The authors are grateful to Anantaram Balakrishnan for his helpful comments on an earlier version of this paper.

TABLE 1 -- USUAL, STRONG, AND PARETO CUTS *

NAME CUT TYPE	#NODES #ITER	#ARCS #ITERI	#DARCS #CUTS	#ODs #FCUTS	CPU	MAS%	OPT VAL UB	 LB	 %DIF	COST ASSUMPTIONS
DATA01	15	60	45	28			8258			
USUAL	73	2	75	4	7135T	??.??	8258	7551	8.56	V = U(50,150)
STRONG	10	2	12	4	10	64.17				F = 10*V
PARETO	7	2	9	4	12	13.33				
DATA02	15	60	45	28			7024			
USUAL	76	3	78	5	7005T	??.??	7024	5963	15.11	V = U(50,150)
STRONG	26	3	28	5	201	95.55				F = 10*V
PARETO	19	3	21	5	125	76.17				
DATA03	15	60	45	28			8444			
USUAL	75	2	76	3	7130T	??.??				V = U(50,250)
STRONG	19	2	20	3	64	88.31				F = U(500,1000)
PARETO	10	2	11	3	23	35.88				
DATA04	15	60	45	28			10071			
USUAL	61	0	61	0	7124T	??.??	10071	7749	23.06	V = U(50,250)
STRONG	25	0	25	0	240	95.80				F = U(500,1000)
PARETO	14	0	14	0	71	65.50				
DATA05	15	60	45	28			9683			
USUAL	66	1	66	1	6792T	??.??	9925	7093	28.53	V = U(100,200)
STRONG	38	1	38	1	829	98.29				F = 1000 - 2*V
PARETO	20	1	20	1	139	76.28				
DATA06	15	60	45	28			10858			
USUAL	55	2	56	3	6860T	??.??	11165	7695	31.08	V = U(100,200)
STRONG	60	2	61	3	6645	99.66				F = 1000 - 2*V
PARETO	26	2	27	3	724	93.81				
DATA07	10	45	35	18			3988			
USUAL	58	1	58	1	2514	99.84				V = U(50,150)
STRONG	12	1	12	1	11	81.86				F = 10*V
PARETO	8	1	8	1	8	24.47				
DATA08	10	45	35	18			4304			
USUAL	33	1	33	1	571	99.59				V = U(50,150)
STRONG	7	1	7	1	3	55.24				F = 10*V
PARETO	7	1	7	1	7	20.06				
DATA09	15	60	45	28			23808			
USUAL	63	2	64	3	7086T	??.??	24046	20738	13.76	V = 10*D + U(0,50)
STRONG	25	2	26	3	170	94.35				F = 2*V + U(250,750)
PARETO	16	1	16	1	73	62.03				
DATA10	15	60	45	28			22259			
USUAL	65	1	65	1	7134	??.??				V = 10*D + U(0,50)
STRONG	44	1	44	1	1453	98.73				F = 2*V + U(250,750)
PARETO	18	1	18	1	102	69.91				

COMPUTATIONS (IN SECONDS) ON A VAX 11/780
T = TIME LIMIT

*EACH GROUP OF ROWS SPECIFIES RESULTS FOR A SINGLE PROBLEM. THE FIRST ROW GIVES THE PROBLEM DESCRIPTION IN THE FORMAT OF THE FIRST ROW IN THE HEADING. THE REMAINING ROWS WITHIN EACH GROUP GIVE THE COMPUTATIONAL RESULTS IN THE FORMAT OF THE SECOND ROW OF THE HEADING.

TABLE 2 -- VARYING NUMBER OF OD PAIRS *

NAME / CUT TYPE	#NODES / #ITER	#ARCS / #ITERI	#DARCS / #CUTS	#ODs / #FCUTS	CPU	MAS%	OPT VAL	UB	LB	%DIF	COST ASSUMPTIONS
DATA11	10	45	35	18			1443				
USUAL	95	0	95	0	7090T	??.??		1465	1289	12.01	V = D
STRONG	23	0	23	0	173	97.60					F = 110 - V
PARETO	13	0	13	0	42	76.07					
DATA11A	10	45	35	45			3152				
USUAL	77	0	77	0	7001T	??.??		3459	2662	23.04	V = D
STRONG	34	0	34	0	1599	99.14					F = 110 - V
PARETO	29	0	29	0	1176	95.36					
DATA12	15	75	45	28			2048				
USUAL	135	0	135	0	6517	99.70					V = D
STRONG	5	0	5	0	4	18.78					F = 110 - V
PARETO	5	0	5	0	11	5.17					
DATA12A	15	75	45	105			6684				
USUAL	74	0	74	0	6985T	??.??		7377	5906	19.94	V = D
STRONG	25	0	25	0	492	90.44					F = 110 - V
PARETO	25	0	25	0	545	67.40					
DATA13	33	100	45	64			5382				
USUAL	126	1	126	1	7115T	??.??		5407	5115	5.40	V = D
STRONG	11	1	11	1	28	25.05					F = 110 - V
PARETO	17	1	17	1	148	14.69					
DATA13A	33	100	45	528			35229				
USUAL	111	1	111	1	7163T	??.??		36001	33683	6.44	V = D
STRONG	55	1	55	1	5290	82.49					F = 110 - V
PARETO	68	0	68	0	7144T	??.??		35238	35171	0.19	

Computations (in seconds) on a CYBER 76
D = Distance Between Points
T = Time Limit

*Each group of rows specifies results for a single problem. The first row gives the problem description in the format of the first row in the heading. The remaining rows within each group give the computational results in the format of the second row of the heading.

TABLE 3 -- BENDERS DECOMPOSITION WITH PREPROCESSING*

NAME	#NODES	#ARCS	#DARCS	#ODs			OPT VAL			
ASCENT			#RARCS	#MCHV	CPU		UB	LB	%DIF	COST
CUT TYPE	#ITER	#ITERI	#CUTS	#FCUTS	CPU	MAS%	UB	LB	%DIF	ASSUMPTIONS
DATA01	15	60	45	28			8258			V = U(50,150)
ASCENT			4	0	14		8258	8258	0.00	F = 10*V
DATA02	15	60	45	28			7024			
ASCENT			7	0	19		7024	6923	1.44	
USUAL	46	3	48	5	81	96.07				V = U(50,150)
STRONG	14	3	16	5	10	73.20				F = 10*V
PARETO	18	3	20	5	36	38.78				
DATA03	15	60	45	28			8444			
ASCENT			6	1	15		8444	8360	0.99	
USUAL	28	2	29	3	24	92.18				V = U(50,250)
STRONG	8	2	10	3	7	54.92				F = U(500,1000)
PARETO	12	2	13	3	19	27.60				
DATA04	15	60	45	28			10071			V = U(50,250)
ASCENT			4	0	15		10071	10071	0.00	F = U(500,1000)
DATA05	15	60	45	28			9683			V = U(100,200)
ASCENT			3	0	14		9683	9683	0.00	F = 1000 - 2*V
DATA06	15	60	45	28			10858			V = U(100,200)
ASCENT			4	0	14		10858	10858	0.00	F = 1000 - 2*V
DATA07	10	45	35	18			3988			V = U(50,150)
ASCENT			2	0	5		3988	3988	0.00	F = 10*V
DATA08	10	45	35	18			4304			V = U(50,150)
ASCENT			2	0	5		4304	4304	0.00	F = 10*V
DATA09	15	60	45	28			23808			V = 10*D + U(0,50)
ASCENT			6	0	15		23808	23808	0.00	F = 2*V + U(250,750)
DATA10	15	60	45	28			22259			V = 10*D + U(0,50)
ASCENT			6	0	14		22259	22259	0.00	F = 2*V + U(250,750)

COMPUTATIONS (IN SECONDS) ON A VAX 11/780

*EACH GROUP OF ROWS SPECIFIES RESULTS FOR A SINGLE PROBLEM. THE FIRST ROW GIVES THE PROBLEM DESCRIPTION IN THE FORMAT OF THE FIRST ROW IN THE HEADING. THE SECOND ROW GIVES THE RESULTS OF THE DUAL ASCENT USING THE FORMAT OF THE SECOND ROW OF THE HEADING. THE REMAINING ROWS, IF NEEDED, USE THE FORMAT OF THE THIRD ROW OF THE HEADING TO GIVE THE COMPUTATIONAL STATISTICS FOR COMPLETING THE OPTIMIZATION USING BENDERS' DECOMPOSITION.

TABLE 4 -- BENDERS DECOMPOSITION WITH PREPROCESSING*

NAME / ASCENT / CUT TYPE	#NODES / #ITER	#ARCS / #ITERI	#DARCS / #RARCS / #CUTS	#DDs / #MCUV / #FCUTS	CPU	MAS%	OPT VAL / UB / UB	LB / LB	%DIF / %DIF	COST ASSUMPTIONS
DATA14	30	130	90	58			19179			
ASCENT			12	3	159		19179	19042	0.71	V = U(50,150)
USUAL	198	3	200	5	5764L	99.19	19179	19172	0.04	F = U(500,1000)
STRONG	17	3	19	5	50	69.87				
PARETO	26	3	28	5	239	33.27				
DATA15	30	130	90	58			17096			V = U(50,150)
ASCENT			7	0	148		17096	17096	0.00	F = U(500,1000)
DATA16	25	120	90	48			12702			
ASCENT			7	1	79		12702	12530	1.35	V = U(50,150)
USUAL	62	0	62	0	170	93.93				F = U(500,1000)
STRONG	15	0	15	0	22	50.81				
PARETO	19	0	19	0	113	18.47				
DATA17	25	120	90	48			15257			V = U(50,150)
ASCENT			6	0	87		15257	15257	0.00	F = U(500,1000)
DATA18	25	120	90	48			13120			V = U(50,150)
ASCENT			5	0	80		13120	13120	0.00	F = U(500,1000)
DATA19	25	120	90	48			13449			
ASCENT			9	1	89		13493	13424	0.51	V = U(50,150)
USUAL	64	2	65	3	198	94.62				F = U(650,850)
STRONG	10	2	11	3	10	41.66				
PARETO	12	2	13	3	58	12.24				
DATA2Ø	25	120	90	48			17484			V = U(50,250)
ASCENT			5	0	76		17484	17484	0.00	F = U(650,850)
DATA21	25	120	90	48			23076			
ASCENT			7	0	86		23076	23049	0.12	V = U(100,200)
USUAL	18	2	20	4	14	79.29				F = 10*V
STRONG	10	2	12	4	9	29.73				
PARETO	10	2	12	4	42	9.08				
DATA22	25	120	90	48			31890			
ASCENT			20	7	90		31890	31270	1.94	V = U(100,200)
USUAL	152	4	155	7	6999T	??.??	31890	31810	0.25	F = 20*V
STRONG	18	5	21	8	48	74.19				
PARETO	23	5	26	8	185	45.97				
DATA23	25	120	90	48			28310			
ASCENT			6	1	88		28310	28117	0.68	V = U(100,200)
USUAL	30	1	30	1	26	81.44				F = 20*V
STRONG	11	1	11	1	11	32.49				
PARETO	16	1	16	1	85	10.50				
DATA24	25	120	90	48			28522			
ASCENT			15	3	93		28522	27211	4.60	V = U(100,200)
USUAL	58	3	60	5	585	98.30				F = 20*V
STRONG	12	3	14	5	17	59.63				
PARETO	13	3	15	5	68	25.39				
DATA25	25	120	90	48			28641			
ASCENT			9	1	87		28861	28316	1.89	V = U(100,200)
USUAL	18	2	20	4	15	79.88				F = 20*V
STRONG	8	2	10	4	7	25.19				
PARETO	12	2	14	4	56	10.57				
DATA26	25	120	90	48			28157			
ASCENT			5	0	84		28157	28077	0.28	V = U(100,200)
USUAL	12	2	13	3	5	65.02				F = 20*V
STRONG	7	2	8	3	5	21.57				
PARETO	8	2	9	3	33	4.09				
DATA27	25	110	90	48			67315			
ASCENT			15	2	94		67315	66721	0.88	V = U(25,325)
USUAL	37	10	51	24	194	97.41				F = 30*V + U(-100,10
STRONG	18	10	32	24	39	80.66				
PARETO	24	9	38	23	176	60.32				

(TABLE CONTINUED)

TABLE 4 (CONTINUED)*

NAME / ASCENT / CUT TYPE	#NODES / #ITER	#ARCS / #ITERI	#DARCS / #PARCS / #CUTS	#ODs / #MCHV / #FCUTS	CPU	MAS%	OPT VAL / UB / UB	LB / LB	%DIF / %DIF	COST ASSUMPTIONS
DATA28	25	120	90	48			44118			V = 10*D + U(50,150)
ASCENT			12	0	113		44118	44118	0.00	F = 2*V + U(-100,100)
DATA29	25	120	90	48			?			
ASCENT			42	8	147		47166	46691	1.01	V = 10*D + U(50,150)
USUAL	131	0	131	0	3169L	98.96	47166	46691	1.01	F = 2*V + U(-100,100)
STRONG	27	0	27	0	1488A	98.22	47166	46691	0.89	
PARETO	28	0	28	0	2106A	91.97	47166	46768	0.84	
DATA30	25	120	90	48			?			
ASCENT			23	2	131		51489	51313	0.34	V = 10*D + U(50,150)
USUAL	199	2	200	3	3495L	98.84	51489	51313	0.34	F = 2*V + U(-100,100)
STRONG	57	2	58	3	4178A	??.??	51489	51356	0.26	
PARETO	51	2	52	3	4484A	94.00	51489	51356	0.26	
DATA31	25	120	90	48			43024			
ASCENT			15	3	87		43024	42897	0.30	V = 10*D + U(50,150)
USUAL	200	1	200	1	6911L	99.48				F = 2*V + U(-100,100)
STRONG	41	1	41	1	395	91.93				
PARETO	57	1	57	1	1122	74.23				
DATA32	25	100	45	48			42388			V = U(25,325)
ASCENT			2	0	96		42388	42388	0.00	F = 20*V + U(-100,100)
DATA33	25	70	45	48			63407			
ASCENT			18	2	92		63407	62254	1.82	V = U(25,325)
USUAL	172	4	176	8	7164T	??.??	63407	62418	1.56	F = 20*V + U(-75,75)
STRONG	29	4	33	8	205	89.16				
PARETO	17	4	21	8	99	51.03				
DATA34	25	70	45	48			43391			V = U(25,325)
ASCENT			9	0	68		43391	43391	0.00	F = 3000 - 10*V - U(0,150)
DATA35	25	105	90	48			?			
ASCENT			90	11	368		17381	16420	5.53	V = U(30,90)
USUAL	74	20	97	43	2747A	??.??	17381	16420	5.53	F = 10*V + U(100,200)
STRONG	43	33	75	65	7071T	??.??	17381	16420	5.53	
PARETO	51	39	82	80	7087T	??.??	17381	16420	5.53	
DATA36	25	120	90	48			?			
ASCENT			21	4	127		39422	39131	0.74	V = U(60,140)
USUAL	160	2	161	3	7103T	??.??	39422	39131	0.74	F = 1000 - V + U(-50,50)
STRONG	71	2	72	3	7031T	??.??	39422	39131	0.74	
PARETO	70	2	71	3	7141T	??.??	39422	39131	0.74	
DATA37	25	120	90	48			?			
ASCENT			45	21	137		19115	18677	2.29	V = D + U(50,150)
USUAL	128	3	130	5	7171T	??.??	19115	18677	2.29	F = 5*V + 5*D
STRONG	67	3	69	5	7008T	??.??	19115	18791	1.70	
PARETO	48	2	50	4	7051T	??.??	19115	18840	1.44	

COMPUTATIONS (IN SECONDS) ON A VAX 11/780
A = SYSTEM TERMINATED ABNORMALLY
L = SYSTEM LOOPED
T = TIME LIMIT

*EACH GROUP OF ROWS SPECIFIES RESULTS FOR A SINGLE PROBLEM. THE FIRST ROW GIVES THE PROBLEM DESCRIPTION IN THE FORMAT OF THE FIRST ROW IN THE HEADING. THE SECOND ROW GIVES THE RESULTS OF THE DUAL ASCENT USING THE FORMAT OF THE SECOND ROW OF THE HEADING. THE REMAINING ROWS, IF NEEDED, USE THE FORMAT OF THE THIRD ROW OF THE HEADING TO GIVE THE COMPUTATIONAL STATISTICS FOR COMPLETING THE OPTIMIZATION USING BENDERS' DECOMPOSITION.

REFERENCES

1. E. Balas and N. Christofides, "A restricted Lagrangean approach to the traveling salesman problem", Mathematical Programming 21 (1981) 19-46.

2. I. Barany, T. Van Roy and L. Wolsey, "Strong formulations for multi-item capacitated lot-sizing", CORE Discussion Paper 8312, Universite Catholique de Louvain (Louvain-la-Neuve, Belgium, March 1983).

3. E. M. L. Beale and J. A. Tomlin, "An integer programming approach to a class of combinatorial problems", Mathematical Programming 3 (1972) 339-344.

4. O. Bilde and J. Krarup, "Sharp lower bounds and efficient algorithms for the simple plant location problem", Annals of Discrete Mathematics 1 (1977) 79-97.

5. J. Billheimer and P. Gray, "Network design with fixed and variable cost elements", Transportation Science 7 (1973) 49-74.

6. F. Boesch, Large-scale networks: theory and design (IEEE Press, 1975).

7. T. B. Boffey and A. I. Hinxman, "Solving the optimal network problem", European Journal of Operational Research 3 (1979) 386-393.

8. D. E. Boyce, ed., Transportation Research B 13B(1) (1979) 1-3.

9. D. E. Boyce, A. Farhi and R. Weischedel, "Optimal network problem: A branch-and-bound algorithm", Environment and Planning 5 (1973) 519-533.

10. C. Cornuejols, M. L. Fisher and G. L. Nemhauser, "Location of bank accounts to optimize flow", Management Science 23 (1977) 789-810.

11. H. Crowder, E. Johnson and M. W. Padberg, "Solving large-scale zero-one linear programming problems", Operations Research 31 (1983) 803-834.

12. H. Crowder and M. W. Padberg, "Large scale symmetric travelling salesman problems", Management Science 26 (1980) 495-509.

13. P. S. Davis and T. L. Ray, "A branch-bound algorithm for capacitated facilities location problems", Naval Research Logistics Quarterly 16 (1969) 331-344.

14. R. Dionne and M. Florian, "Exact and approximate algorithms for optimal network design", Networks 9 (1979) 37-59.

15. D. Erlenkotter, "A dual based procedure for uncapacitated facility location", Operations Research 26 (1978)992-1009.

16. M. L. Fisher, R. Jaikumar and L. VanWassenhove, "A multiplier adjustment method for the generalized assignment problem" (presented at ORSA/TIMS Meeting, Washington, D.C., May 1980).

17. M. L. Fisher and D. S. Hochbaum, "Data base location in computer networks", Journal of the ACM 7 (1980) 718-735.

18. M. Florian, G. Guerin and G. Bushel, "The engine scheduling problem in a railway network", INFOR Journal 14 (1976) 121-138.

19. R. Francis and J. White, Facility layout and locations, an analytical approach (Prentice-Hall, Englewood Cliffs, New Jersey, 1974).

20. G. Gallo, "A new branch-and-bound algorithm for the network design problem", Report L81-1, Instituto Di Elaborazione Dell' Informazione (Pisa, Italy, 1981).

21. G. Gallo, "Lower planes for the network design problem", Networks 13 (1983) 411-426.

22. A. M. Geoffrion and G. Graves, "Multicommodity distribution system design by Benders decomposition", Management Science 5 (1974) 822-844.

23. A. Geoffrion and R. Marsten, "Integer programming algorithms: a framework and state-of-the-art survey", Management Science 18 (1972) 465-491.

24. S. Graves and B. Lamar, "An integer programming procedure for assembly system design problems", Operations Research 31 (1983) 522-545.

25. M. Guignard, "Preprocessing and optimization in network flow problems with fixed charges", Department of Statistics, Report #47, Wharton School, University of Pennsylvania (1982).

26. M. Guignard and K. Spielberg, "A direct dual method for the mixed plant location problem with some side constraints", Mathematical Programming 17 (1979) 198-228.

27. M. Guignard and K. Spielberg, "Logical reduction methods in 0-1 programming", Operations Research 29 (1981) 49-74.

28. G. Handler and P. Mirchandani, Location on networks: theory and algorithms (MIT Press, Cambridge, Mass., 1979).

29. H. H. Hoang, "A computational approach to the selection of an optimal network", Management Science 19 (1973) 488-498.

30. D. S. Johnson, J. K. Lenstra and A. H. G. Rinnooy Kan, "The complexity of the network design problem", Networks 8 (1978) 279-285.

31. J. L. Kennington and R. V. Helgason, Algorithms for network programming (John Wiley and Sons, 1980).

32. D. Knuth, The art of computer programming: Volume 2, seminumerical algorithms (Addison-Wesley, 1969).

33. J. Krimenia and S. B. Gershwin, "Network flow optimization in flexible manufacturing systems", Proceedings of 1978 IEEE Conference on Decision and Control (1979) 633-639.

34. A. Land and S. Powell, Fortran codes for mathematical programming (John Wiley and Sons, London, 1973).

35. A. Land and S. Powell, "Computer codes for problems of integer programming", Annals of Discrete Mathematics 5 (1979) 221-269.

36. A. Land and S. Powell, "A survey of available computer codes to solve integer linear programming problems", Research Report No. 81-09, Ecole des Hautes Etudes commerciales, University of Montreal (April 1981).

37. R. C. Larson and A. R. Odoni, Urban operations research (Prentice-Hall, Englewood Cliffs, New Jersey, 1981).

38. C. E. Lemke and K. Spielberg, "Direct search zero one and mixed integer programming", Operations Research 15 (1967) 892-914.

39. M. Los and C. Lardinois, "Combinatorial programming, statistical optimization and the optimal transportation network problem", Transportation Research-B 16B (1982) 89-124.

40. T. L. Magnanti, "Combinatorial optimization and vehicle fleet planning: Perspectives and prospects", Networks 11 (1981) 179-214.

41. T. L. Magnanti and R. T. Wong, "Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria", Operations Research 29 (1981) 464-484.

42. T. L. Magnanti and R. T. Wong, "Network design and transportation planning: Models and algorithms", Transportation Science 18 (1984a) 1-56.

43. T. L. Magnanti and R. T. Wong, "Decomposition methods for facility location problems" (1984b).

44. T. L. Magnanti and R. T. Wong, "A dual-ascent approach for network design problems", in preparation (1984c).

45. T. G. Mairs, G. W. Wakefield, E. L. Johnson and K. Spielberg, "On a production allocation and distribution problem", Management Science 24 (1978) 1622-1630.

46. C. Mandl, "A survey of mathematical optimization models and algorithms for designing and extending irrigation and wastewater networks", Water Resource Research 17(1) (1981).

47. K. Martin and L. Schrage, "Subset coefficient reduction cutting planes for 0/1 mixed-integer programming", Graduate School of Business, University of Chicago (Revised October 1982).

48. G. Nemhauser and L. Wolsey, "Maximizing submodular set functions: Formulation and analysis of algorithms," Annals of Discrete Mathematics 11 (19) 279-301.

49. R. L. Rardin and U. Choe, "Tighter relaxations of fixed charge network flow problems", Technical Report, No. J-79-18, School of Industrial and Systems Engineering, Georgia Institute of Technology (Atlanta, GA, May 1979).

50. R. Richardson, "An optimization approach to routing aircraft", Transportation Science 10 (1976) 52-71.

51. R. Simpson, "Scheduling and routing models for airline systems", MIT Flight Transportation Laboratory Report (December 1969).

52. B. C. Tansel, R. L. Francis and T. L. Lowe, "Location on networks: A survey, parts I and II", Management Science 29 (1983) 482-511.

53. T. J. Van Roy and D. Erlenkotter, "A dual-based procedure for dynamic facility location", Management Science 28 (1982) 1091-1105.

54. H. P. Williams, "Experiments in the formulation of integer programming problems", Mathematical Programming Study 2 (1974) 180-197.

55. R. T. Wong, "Probabilistic analysis of network design problem heuristic", Krannert Graduate School of Management, Purdue University (West Lafayette, Indiana, 1984).

56. R. T. Wong, "Worst-case analysis of network design problem heuristics", SIAM Journal on Algebraic Discrete Methods 1 (1980) 41-63.

57. R. T. Wong, "A dual ascent approach for Steiner tree problems on a directed graph", Mathematical Programming 28 (1984) 271-287.

58. R. T. Wong, "Accelerating Benders decomposition for network design", Doctoral Dissertation, Department of Electrical Engineering and Computer Science, MIT (February 1978).

APPENDIX

THIS APPENDIX CONTAINS DATA FOR THE FIVE TEST PROBLEMS (29, 30, 35, 36, AND 37) THAT THE ALGORITHMS DESCRIBED IN THIS PAPER DID NOT SOLVE TO OPTIMALITY.

TABLE A.1 DATA FOR PROBLEM DATA29

$V = 10*D + U(50,150)$
$F = 2*V + U(-100,100)$
$D \leq 50$
NODES = 25
#ARCS = 120

ARC NODES		VARIABLE COST	FIXED COST
1	3	418	909
1	4	263	494
1	5	469	919
1	6	298	584
1	8	444	0
1	10	483	1042
1	11	465	839
2	3	460	849
2	4	273	483
2	5	477	0
2	9	385	738
2	10	474	1013
3	4	272	616
3	5	223	346
3	10	448	0
3	12	524	0
4	5	307	0
4	6	180	0
4	7	353	729
4	8	256	611
4	9	228	415
4	10	309	519
4	11	346	0
4	12	448	840
4	14	573	0
4	16	563	1189
5	6	321	659
5	7	190	326
5	8	216	408
5	11	370	750
5	12	344	632
5	15	454	891
5	17	510	936
6	7	421	804
6	9	208	358
6	10	275	590
6	12	439	882
6	14	421	885
6	15	535	1142
7	8	271	631
7	9	362	700
7	10	318	651
7	16	491	0
7	17	501	906
7	21	609	1245
8	9	224	0
8	11	291	0
8	18	479	961
9	10	243	438
9	12	382	727
9	13	333	641
9	14	359	0
9	17	474	883
9	18	482	970
9	19	518	996
9	20	537	0
9	21	612	1254
10	11	200	334
10	13	383	770
10	15	314	0

ARC NODES		VARIABLE COST	FIXED COST
10	17	293	634
10	18	432	953
10	20	546	1119
10	21	543	0
10	23	559	0
11	12	342	0
11	15	336	729
11	17	369	0
11	19	424	841
11	23	512	1009
11	24	587	1082
12	13	569	0
12	14	489	995
12	15	305	606
12	17	293	532
12	18	320	614
12	19	488	0
12	25	607	1309
13	18	606	1256
13	19	363	0
13	22	467	1011
14	15	419	856
14	16	483	1006
14	18	529	1053
14	21	536	978
14	22	359	762
14	23	539	1156
14	24	417	0
14	25	565	1226
15	16	225	522
15	20	482	965
15	22	358	0
15	23	304	593
15	24	439	0
16	17	187	434
16	18	158	341
16	19	372	824
16	22	442	0
16	23	351	0
16	25	468	1006
17	21	301	0
17	24	394	713
18	19	374	778
18	20	550	0
18	21	203	309
18	22	424	821
18	23	237	425
18	24	426	841
18	25	372	679
19	20	266	0
19	21	360	817
19	22	176	388
19	25	392	0
20	25	494	932
21	22	382	669
21	23	162	349
21	24	393	783
22	25	337	664
23	24	291	483
23	25	247	544

TABLE A.2 DATA FOR PROBLEM DATA30

$V = 10*D + U(50,150)$
$F = 2*V + U(-100,100)$
$D \leq 50$
NODES = 25
#ARCS = 125

ARC NODES		VARIABLE COST	FIXED COST
1	2	129	209
1	5	278	573
1	12	581	0
1	13	597	1216
2	3	239	450
2	4	482	948
2	5	257	0
2	7	499	1056
2	12	585	1235
2	13	576	0
3	5	303	530
3	7	376	841
3	8	468	1033
3	9	573	1069
3	10	432	873
3	11	444	0
3	12	539	990
4	7	487	946
4	8	478	860
4	10	419	0
4	11	520	1070
4	14	549	1187
5	6	365	0
5	7	508	1089
5	9	413	903
5	12	455	892
5	16	557	1152
6	9	377	755
6	10	274	470
6	13	334	649
6	14	425	797
6	15	540	1146
6	17	464	892
6	18	483	867
6	19	515	998
7	10	193	327
7	12	259	474
7	13	419	910
7	14	377	659
7	18	364	655
7	21	606	0
8	9	434	0
8	12	309	0
8	15	342	0
8	17	402	0
8	18	352	0
8	19	449	914
8	20	540	1009
8	21	593	0
8	22	592	1201
9	10	453	962
9	11	383	689
9	12	212	0
9	13	122	216
9	14	559	0
9	16	245	416
9	17	395	744
9	18	359	0
9	19	466	900
9	22	637	0
9	24	593	1126
10	11	119	306
10	12	313	680
10	14	255	539
10	16	410	915
10	21	547	0
11	12	367	748
11	13	419	851
11	14	248	539
11	17	313	671
11	22	518	1054
12	15	538	1164
12	17	387	757
12	18	274	636
12	20	556	0
12	21	497	1086
12	22	525	0
12	23	506	1039
12	24	572	1054
13	14	568	0
13	15	623	0
13	16	289	663
13	17	407	0
13	18	342	0
13	19	450	0
14	16	469	841
14	18	365	737
14	22	489	1003
14	24	510	981
14	25	516	997
15	16	522	0
15	17	308	672
15	20	268	531
15	21	521	999
15	22	436	802
15	23	590	0
15	24	530	1016
16	18	191	340
16	21	342	615
16	22	437	814
16	23	336	683
16	25	637	1227
17	18	155	361
17	20	311	678
17	22	368	797
17	23	340	775
17	24	352	753
17	25	453	965
18	19	286	635
18	20	390	0
18	22	299	559
18	23	345	748
18	25	552	1089
19	23	307	596
20	23	478	0
20	24	446	798
21	23	90	226
22	25	287	504
23	24	95	0
24	25	401	797

TABLE A.3 DATA FOR PROBLEM DATA35

V = U(30,90)
F = 10*V + U(100,200)
# NODES = 25
# ARCS = 105

ARC NODES		VARIABLE COST	FIXED COST
1	2	39	578
1	4	61	0
1	5	86	1023
1	6	72	900
1	7	64	803
1	9	60	707
2	4	46	0
2	5	66	806
2	6	45	629
2	7	47	0
2	8	85	1015
2	9	81	923
3	4	57	686
3	5	49	600
3	6	65	785
3	7	83	0
3	8	59	756
3	10	74	911
3	11	77	911
4	5	68	856
4	6	50	0
4	7	56	745
4	8	67	783
4	9	66	803
5	6	57	766
5	7	63	812
5	8	48	582
5	10	81	955
5	11	60	711
6	7	40	555
6	8	64	761
6	9	72	845
7	8	60	700
7	9	50	610
7	13	81	1004
8	10	59	699
8	11	61	779
8	12	74	844
9	13	60	0
9	16	79	939
9	17	77	967
10	11	52	715
10	12	53	0
10	14	80	966
10	15	69	875
11	12	56	738
11	14	68	800
11	15	70	895
12	14	73	0
12	15	72	857
12	18	80	946
12	19	67	797
13	14	84	1023
13	16	52	703
13	17	67	771
13	18	53	684
13	20	69	864
13	21	67	770
13	22	66	839
13	24	73	843
14	15	65	821
14	18	52	707
14	19	76	0
14	20	56	687
14	21	47	589
14	22	71	815
14	23	78	975
14	24	80	954
14	25	64	782
15	18	72	837
15	19	59	757
15	21	71	836
15	23	43	0
15	25	59	725
16	17	52	639
16	18	70	0
16	20	50	0
16	21	77	933
16	22	71	0
16	24	60	774
17	18	76	893
17	20	54	677
17	21	63	746
17	22	61	797
17	24	50	641
18	19	70	874
18	20	58	0
18	21	50	0
18	22	47	654
18	24	66	775
18	25	58	764
19	21	70	896
19	23	34	527
19	25	70	863
20	21	65	779
20	22	38	544
20	24	41	572
21	22	48	615
21	23	65	808
21	24	65	847
21	25	63	782
22	24	49	590
22	25	85	982
23	25	56	668
24	25	75	923

TABLE A.4 DATA FOR PROBLEM DATA36

V = U(60,140)
F = 1000 - V + U(100,200)
# NODES = 25
#ARCS = 125

ARC NODES		VARIABLE COST	FIXED COST
1	3	128	0
1	6	257	810
2	3	61	955
2	4	221	0
2	5	217	875
2	6	251	845
3	4	213	877
3	6	202	0
3	7	348	676
4	6	167	846
4	7	211	799
4	8	273	813
4	9	386	634
4	10	393	619
5	8	262	809
6	7	140	876
6	8	220	855
6	9	330	715
6	10	255	771
6	11	279	792
6	13	380	652
6	14	376	0
7	8	82	979
7	12	298	763
7	13	282	0
7	14	365	718
7	15	317	708
7	16	336	0
8	9	106	0
8	11	211	839
8	13	242	0
8	15	273	790
8	16	313	0
8	17	366	0
9	10	241	855
9	11	230	776
9	12	153	866
9	13	205	842
9	16	298	778
9	17	290	710
9	19	382	675
10	14	159	854
10	16	151	881
10	17	260	773
10	18	270	0
10	19	257	772
10	20	344	722
10	21	371	692
11	12	218	0
11	15	176	861
11	16	134	915
11	19	239	781
11	20	322	690
11	21	362	655
11	22	403	0
11	24	398	628
12	13	103	996
12	15	80	958
12	17	143	939
12	18	388	691
12	19	267	787
12	20	211	847
12	22	332	0
12	23	366	717
13	14	222	859
13	16	96	0
13	17	129	955
13	20	202	847
13	21	357	651
13	22	318	715
13	23	327	0
13	24	344	0
13	25	360	715
14	15	257	0
14	16	144	913
14	17	266	760
14	18	128	962
14	22	366	693
14	24	295	0
15	16	112	902
15	18	303	739
15	19	184	0
15	20	178	914
15	21	354	738
15	22	286	0
15	25	334	0
16	17	134	920
16	18	193	817
16	19	104	897
16	20	202	0
16	21	264	804
16	22	271	751
16	23	268	802
16	24	272	794
17	19	149	876
17	20	89	966
17	21	300	0
17	22	195	825
17	23	218	867
17	24	262	758
18	19	146	895
18	21	113	971
18	22	302	743
18	23	242	0
18	24	194	0
19	21	177	832
19	23	166	0
19	24	165	847
19	25	277	749
20	21	314	742
20	22	127	934
20	23	171	870
20	24	240	0
20	25	163	0
21	22	253	801
21	23	188	888
21	24	100	952
22	23	89	0
22	24	165	911
24	25	230	855

TABLE A.5 DATA FOR PROBLEM DATA37

$V = D + U(50,150)$
$F = 5*V + 5*D$
$D \leq 50$
NODES = 25
ARCS = 120

ARC NODES		VARIABLE COST	FIXED COST
1	2	117	715
1	5	79	450
1	6	167	0
1	7	90	590
1	8	91	0
1	9	128	820
2	3	114	0
2	6	107	590
2	7	154	850
2	8	98	610
2	10	143	935
2	11	150	990
3	5	186	1125
3	6	129	685
3	7	72	440
3	8	81	510
3	11	128	870
4	5	72	370
4	6	96	0
4	8	159	0
4	10	151	950
4	11	125	860
5	6	144	895
5	7	79	535
5	8	144	0
5	9	129	775
5	10	170	1040
5	13	134	890
6	9	97	660
6	11	140	890
6	12	99	695
7	8	60	345
7	9	169	0
7	10	145	860
7	12	123	0
7	13	156	990
8	10	150	865
8	12	109	680
8	13	141	900
8	16	125	875
9	13	97	0
9	14	112	740
9	15	165	970
9	18	90	640
9	20	116	810
9	22	153	0
10	11	120	640
10	12	86	0
10	13	145	0
10	14	101	615
10	17	157	915
10	19	181	0
10	21	126	820
10	23	126	850
10	25	130	900
11	12	115	600
11	13	102	610
11	14	136	0
11	15	107	690
11	16	175	0
11	18	119	720
11	19	124	775
11	21	171	0
11	25	160	1020
12	13	164	940
12	14	89	495
12	16	88	590
12	17	129	755
12	20	168	965
12	22	126	830
12	23	150	0
13	14	182	1080
13	15	69	405
13	16	95	530
13	18	148	0
13	19	138	910
13	20	159	955
13	22	158	950
13	23	148	950
14	15	108	755
14	17	151	895
14	18	91	610
14	19	83	0
14	22	163	1045
14	23	129	805
14	24	146	0
14	25	144	0
15	19	181	0
15	20	108	710
15	25	115	780
16	17	86	505
16	18	109	620
16	19	113	0
16	20	173	1005
16	21	123	730
16	22	81	0
16	23	115	0
17	19	117	720
17	20	124	685
17	22	126	720
17	23	76	495
17	25	115	700
18	19	137	0
18	20	138	755
18	21	140	745
18	24	152	915
19	21	123	745
19	22	149	0
19	24	134	750
19	25	114	700
20	23	157	835
20	24	105	0
20	25	92	540
21	22	121	665
21	24	147	0
21	25	145	790
22	24	145	890
23	24	101	545
23	25	145	0
24	25	113	640