An Algorithm for the Resource Constrained Shortest Path Problem

J. E. Beasley and N. Christofides
School of Management, Imperial College, London SW7 2AZ, England

In this paper we examine an integer programming formulation of the resource constrained shortest path problem. This is the problem of a traveller with a budget of various resources who has to reach a given destination as quickly as possible within the resource constraints imposed by his budget. A lagrangean relaxation of the integer programming formulation of the problem into a minimum cost network flow problem (which in certain circumstances reduces to an unconstrained shortest path problem) is developed which provides a lower bound for use in a tree search procedure. Problem reduction tests based on both the original problem and this lagrangean relaxation are given. Computational results are presented for the solution of problems involving up to 500 vertices, 5000 arcs, and 10 resources.

I. INTRODUCTION

The resource constrained shortest path problem is the problem of finding the shortest path between two vertices on a network whenever the traversal of any arc/vertex consumes certain resources and the resources consumed along the path chosen must lie within given limits (both lower and upper limits). The problem can be likened to that of a traveller with a budget (a multidimensional vector of resources) who has to reach a given destination as quickly as possible within the constraints imposed by his budget.

Constrained shortest path problems have been considered by a number of authors in the literature and we can identify three basic types of problem:

(1) the resource constrained shortest path problem (RCSP) as defined above;

(2) the vertex constrained shortest path problem (VCSP) which is the problem of finding the shortest path constrained to pass through a set of specified vertices (or a number of vertices chosen from specified subsets);

(3) the time constrained shortest path problem (TCSP) where the lengths of the arcs are time-dependent and/or time windows exist for the vertices.

TCSP is outside the scope of this paper but the interested reader is referred to [9,11,13,14,22].

Note here that VCSP can be viewed as a special case of RCSP (e.g., simply associate

NETWORKS, Vol. 19 (1989) 379–394

CCC 0028-3045/89/030379-16$04.00

a different resource with each vertex in the set of specified vertices through which the shortest path must pass and impose appropriate resource limits). We identify it separately here in order to help classify the literature on the constrained shortest path problem. (For a more extensive classification of the literature on shortest path problems see Deo and Pang [10]).

Note also that the (open) travelling salesman problem (TSP) is an important special case of VCSP. However, the literature concerning the TSP is extensive and, in order to achieve clarity and brevity, we neglect such work in our literature survey given below. The reader interested in the TSP is referred to Lawler, Lenstra, Rinnooy Kan, and Shmoys [37].

A. Literature Survey

(1) VCSP

Kalaba [33] discussed VCSP and a solution algorithm based upon dynamic programming was proposed by Saksena and Kumar [48]. Their algorithm is incorrect, as noted by Dreyfus [16]. Bajaj [3] presented an algorithm for VCSP based upon dynamic programming. Ibaraki [29] presented two algorithms, one based upon dynamic programming and the other based upon tree search. Kershenbaum, Hsieh, and Golden [35] also presented an algorithm based upon dynamic programming. See also Mori and Nishimura [41] (from [10]).

(2) RCSP

The literature on RCSP can be conveniently classified into four types:

(a) work dealing with a single resource and a single inequality constraint (imposing an upper limit on the total amount of resource consumed along the chosen path);

(b) work dealing with a single resource and two inequality constraints (imposing upper and lower limits on the total amount of resource consumed along the chosen path);

(c) work, drawn from the literature on multiple objective problems, concerned with generating "efficient" paths;

(d) work dealing with the general (many resources) problem.

We consider each type in turn.

(a) Early work dealing with RCSP in the case of a single resource and a single inequality (upper limit) was presented by Witzgall and Goldman [55]. Joksch [32], working independently, presented a similar algorithm to Witzgall and Goldman based upon dynamic programming (see also Lawler [36]).

Aneja and Nair [2] presented an algorithm for the problem based upon a parametric approach but no computational results were given. Note here that their algorithm is incorrect (see [1,43]).

Handler and Zang [23] presented an algorithm based upon a lagrangean relaxation of the problem (in essence the same relaxation as we present in this paper), solved as an unconstrained shortest path problem. As there is only a single lagrange multiplier the optimal value of the lagrangean relaxation can be found very easily. The optimal

solution to the original problem is then found by applying an algorithm that successively enumerates all shortest paths (in increasing length order) in the network with arc lengths modified by the lagrange multipliers. They report solving problems with 500 vertices and 2500 arcs in 20 seconds on a CDC 6600.

Jaffe [30] presented an analysis of the problem of deciding whether a path satisfying the resource constraint and of length at most L exists or not (a NP-complete problem (see [21])).

(b) Early work dealing with RCSP in the case of a single resource and two inequalities (upper and lower limits) was presented by Saigal [47] who considered the problem of finding the shortest path containing exactly q arcs and developed a dynamic programming recursion (but see [46]). Note here that such "q-paths" have been used by Houck, Picard, Queyranne, and Vemuganti [28] to generate a lower bound for the travelling salesman problem and by Christofides, Mingozzi, and Toth [6] and Beasley [4] to generate a lower bound for the vehicle routing problem.

Ribeiro and Minoux [44,45] presented two algorithms for the problem, the first [44] being a heuristic based upon generating efficient solutions via parametric shortest path calculations and the second [45] being based upon lagrangean relaxation. They report solving problems (derived from equality constrained knapsack problems [40]) with up to 50,000 vertices and 100,000 arcs.

(c) For a problem involving K resources each path P can be regarded as having $(K + 1)$ scalar parameters, a length L and a resource usage $R_k (k = 1, \ldots, K)$, and can be denoted by $P(L,R_1, \ldots, R_K)$. The path $P(L,R_1, \ldots, R_K)$ is "efficient" if and only if it is impossible to find a path $P_1(L_1,r_1, \ldots, r_K)$ with $L_1 \leq L$ and $r_k \leq R_k (k = 1, \ldots, K)$ and either $L_1 < L$ or $r_k < R_k$ for some $k (1 \leq k \leq K)$. In other words a path is efficient if we cannot decrease one parameter without increasing at least one other parameter. Note here that an efficient path is often called a Pareto optimal path.

The problem of generating efficient paths in the case of a single resource is known as the bicriterion path problem and has been considered by a number of authors in the literature. Vincke [51] presented an algorithm based upon the shortest path algorithm of Ford [20] (from [24]). Hansen [24] presented an algorithm based upon labeling. Climaco and Martins [8] presented an algorithm based upon enumerating paths in increasing length order and report solving problems involving up to 500 vertices and 2567 arcs. Henig [26] presented a theoretical paper dealing with the problem.

The problem of generating efficient paths with multiple resources has been considered by White [53,54] who presented a number of algorithms. Loui [38] considered the problem of generating efficient paths in order to find a utility maximizing path and presented an algorithm based upon extending unconstrained shortest path algorithms. Martins [39] presented two algorithms, one of which being a generalization of the labeling algorithm of Hansen [24]. Warburton [52] presented an algorithm based upon approximating efficient sets to any desired accuracy.

(d) Early work dealing with the general (many resources) problem was presented by Joksch [32] who mentioned that the dynamic programming recursion he developed for the single resource case could be extended to problems with more than one resource

(no details were given). Jensen and Berry [31] presented an algorithm based upon dynamic programming and the use of dominance.

Aneja, Aggarwal, and Nair [1] presented an algorithm based upon a number of reduction tests and a generalization of the dynamic programming algorithm for the unconstrained shortest path problem due to Dijkstra [15]. No computational results were given.

Desrochers [12] considered a generalization of RCSP in which each vertex had an associated "resource window" and the resource usage on the path (if any) to a vertex has to fall within the limits specified by the resource window. He presented a primal-dual algorithm based upon dynamic programming and reported solving problems involving up to 1000 vertices, 52,978 arcs, and 5 resources. Note here that a resource window can be regarded as a generalization of the concept of a time window to two (or more) dimensions.

B. Discussion

One approach to solving RCSP (and VCSP) is by use of an algorithm which lists all the paths between the origin and destination vertices in increasing length order; then the first path found which satisfies the resource constraints will be the optimal solution. Such algorithms are available both for producing elementary paths (Yen [57]) and for nonelementary paths (Hoffman and Pavley [27], Bellman and Kalaba [5]). See [34,42,49,50] for recent work on this problem. (Note here that this approach can also be used to generate efficient paths.)

The difficulty with this approach to RCSP is that no feasible solution is found until the optimal solution is found and an enormous number of paths may have to be generated and rejected before this occurs. This approach has been investigated for the single resource problem by Handler and Zang [23], who found it to be expensive computationally and not competitive with a special purpose algorithm such as they developed.

Another approach to solving RCSP with upper resource limits is to generate efficient paths and then choose from these paths the least cost alternative which satisfies the resource constraints. We would expect that an approach of this kind would not be computationally competitive with a special purpose algorithm for the problem.

In this paper we show how RCSP can be formulated as an integer program and develop a tree search procedure for the problem using a lower bound derived from a lagrangean relaxation of this program. The use of lagrangean relaxation for RCSP has been examined by Chuah [7] who found that for small problems optimal solutions could often by obtained quickly by trying different values for the lagrange multipliers. In this paper we extend his work by incorporating the lower bound into a tree search procedure, by developing problem reduction tests and by examining the performance of the procedure on larger problems (up to 500 vertices, 5000 arcs and 10 resources).

Note here that although the algorithm for RCSP presented in this paper deals with problems which involve both lower and upper resource limits computational results are only presented for problems involving upper limits.

II. PROBLEM FORMULATIONS

Consider the network $G = (V,A)$, where V is a set of n vertices and A is a set of m arcs. We will write c_{ij} for the length of arc (i,j), where $c_{ij} = \infty$ if $(i,j) \notin A$ and

$c_{ii} = 0 \; \forall \; i \in V$. The origin for the path is taken as vertex 1 and the destination as vertex n.

A. Dynamic Programming Formulation

RCSP can be formulated as a dynamic program as follows:
Let

K	be the number of resources we are considering
$\mathbf{q}_i$	represent the vector of resources consumed in passing through vertex i ($\mathbf{q}_i \geqslant \mathbf{O} \; \forall i \in V$ and, without loss of generality, assume $\mathbf{q}_1 = \mathbf{q}_n = \mathbf{O}$)
$\mathbf{r}_{ij}$	represent the vector of resources consumed in traversing arc $(i,j) \in A$ ($\mathbf{r}_{ii} = \mathbf{O} \; \forall \; i \in V$, $\mathbf{r}_{ij} \geqslant \mathbf{O} \; \forall \; (i,j) \in A$)
$\mathbf{L}$	represent the vector of resources that must be consumed on the chosen path (i.e., a lower limit on the resources consumed on the path), where $\mathbf{L} \geqslant \mathbf{O}$
$\mathbf{U}$	represent the vector of resources corresponding to the maximum amount that can be used on the chosen path (i.e., an upper limit on the resources consumed on the path) where $\mathbf{U} \geqslant \mathbf{L} \geqslant \mathbf{O}$
$f(i, \mathbf{R})$	represent the length of the shortest path from vertex 1 to vertex i involving consumption of resource $\mathbf{R}$.

Then the dynamic programming recursion is given by

$$f(i,\mathbf{R}) = \min \left[f(j, \mathbf{R} - \mathbf{r}_{ji} - \mathbf{q}_i) + c_{ji} \mid (j,i) \in A \; \forall \; j \in V \right] \tag{1}$$

and the optimal path has length

$$\min \left[f(n,\mathbf{R}) \mid \mathbf{L} \leqslant \mathbf{R} \leqslant \mathbf{U} \; \forall \; \mathbf{R} \right] \tag{2}$$

Note here that the algorithm for RCSP presented by Aneja, Aggarwal, and Nair [1] is based upon the dynamic programming recursion given above.

Consider now the nature of the (resource constrained) shortest path as calculated by Equations (1) and (2). It is clear that this path is not guaranteed to be either (i) elementary (no vertex revisited); or (ii) simple (no arc repeated).

However we can (for a general network G) *guarantee* that the solution to RCSP is elementary (and hence simple) if

(a) the network G contains no negative cost circuits; *and*

(b) the constraint specifying the lower limit on the resources consumed on the path chosen is redundant (i.e., removing this constraint by setting $\mathbf{L} = \mathbf{O}$ leaves the optimal path unchanged).

Any network G for which both conditions (a) and (b) are satisfied we shall call *g-elementary* (indicating that the solution to RCSP is guaranteed to be elementary). Note here that a network G in which the solution to RCSP is elementary is not necessarily g-elementary.

While condition (a) can be checked (for any network) in $O(n^3)$ operations by applying Floyd's [19] algorithm to the matrix (c_{ij}) it is clear that, for a general network, condition

(b) is difficult to check without actually solving RCSP. Note here that any network in which $c_{ij} \geq 0 \; \forall \; (i,j) \in A$ and $\mathbf{L} = \mathbf{O}$ is g-elementary.

If we require the solution to RCSP to be elementary (but cannot guarantee that it will automatically be elementary) then we simply add an extra resource k for each vertex and limit that resource by $L_k = 0$ and $U_k = 1$. The same approach can be used to enforce any set of degree constraints on the vertices in the problem and this is the basis of the dynamic programming algorithms mentioned earlier for VCSP. Note also that a similar approach can be used to limit the number of times any arc in the network is used in the solution to RCSP.

Computationally, the solution of the dynamic programming formulation of RCSP requires $O(mnd)$ operations where d is the number of distinct resource vectors that need to be considered. We would expect that this formulation would only be suitable for solving problems for which d is small.

B. Integer Programming Formulation

RCSP can be formulated as a zero–one integer program as follows:

$$\begin{aligned} \text{Define } x_{ij} &= 1 \quad \text{if arc } (i,j) \in A \text{ is in the optimal path} \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Note here that by defining x_{ij} to be a zero-one variable we are implicitly assuming that we are seeking a solution to RCSP which is simple (no arc repeated). If we wish to allow an arc (i,j) to be used more than once then we would define x_{ij} to be a general integer variable. Henceforth we shall assume that we are seeking a solution to RCSP which is simple. The integer programming formulation of RCSP is then given by

$$\min \quad \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{3}$$

$$\text{s.t. } L_k \leq \sum_{i \in V} \sum_{j \in V} r_{ijk} x_{ij} + \sum_{j \in V} q_{jk} \left(\sum_{i \in V} x_{ij} + \sum_{m \in V} x_{jm} \right) \Big/ 2 \leq U_k \qquad k = 1, \ldots, K \tag{4}$$

$$\sum_{i \in V} x_{ij} = \sum_{i \in V} x_{ji} \qquad \forall \; j \in V \; (j \neq 1, n) \tag{5}$$

$$\sum_{j \in V} x_{1j} = 1 \tag{6}$$

$$\sum_{i \in V} x_{in} = 1 \tag{7}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq M \sum_{i \in S} \sum_{j \in V - S} x_{ij} \qquad \forall \; S \subseteq V - \{n\} \tag{8}$$

$$x_{ij} \in (0,1) \qquad \forall \; i,j \in V \tag{9}$$

Equation (4) ensures that the total amount of any resource k used on the path should lie between L_k and U_k (recall here that we have assumed that $q_{1k} = q_{nk} = 0$, $k = 1, \ldots, K$). Note that this equation can be written in a simpler form but the form used above is preferred since it makes the lagrangean relaxation of the problem (which we develop later) symmetric.

Equation (5) is the degree constraint for each vertex of the network (other than the

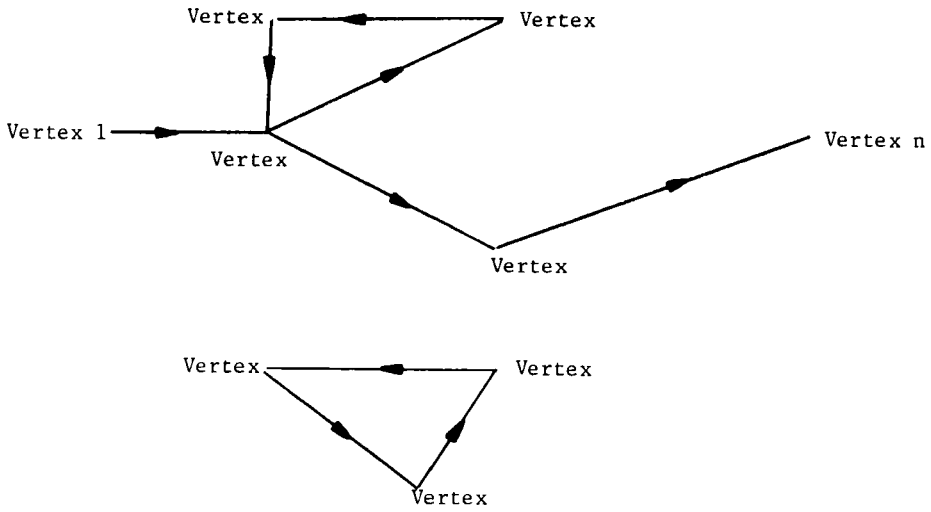

FIG. 1. Example.

origin and destination vertices) while Eqs. (6) and (7) ensure that one arc leaves the origin vertex and one arc enters the destination vertex.

In Eq. (8), M is a large positive constant and the equation ensures that no arc can be used within any subset $S \subseteq V - \{n\}$ of vertices unless there is at least one arc directed out of the subset S. This equation ensures that situations of the kind shown in Figure 1, where we have an isolated group of arcs, cannot occur. Note that there is no requirement on the solution to RCSP to be elementary (no vertex visited more than once)—indeed the path shown in Figure 1 is not elementary. If we wish the solution to RCSP to be elementary then we can simply replace Eq. (8) by the subtour elimination constraint

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad \forall S \subseteq V \tag{10}$$

If the network G is g-elementary then the solution to RCSP is guaranteed to be an elementary path and Eqs. (8) and (10) are redundant.

Note here that in the case of the solution to RCSP being allowed to be nonelementary we cannot assume that only one arc leaves the origin vertex (as Eq. (6)). However, this difficulty can be overcome by introducing an artificial origin vertex connected by just one arc to the original origin vertex. A similar approach can be adopted for the destination vertex (Eq. [7]). Henceforth we shall assume that the origin and destination vertices have been defined in this way.

In the next section we develop a lower bound, for use in a tree search procedure, from a lagrangean relaxation of the integer programming formulation of the problem given above.

III. THE LOWER BOUND

To derive a lower bound for RCSP we drop Eq. (8) from the problem (Eq. (10) if we are seeking an elementary solution to RCSP) and relax the resource constraint (Eq.

(4)) in a lagrangean fashion. Let $s_k \geq 0$ represent the lagrange multiplier for the left-hand inequality in Eq. (4) and $t_k \geq 0$ the lagrange multiplier for the right-hand inequality in Eq. (4). The lagrangean lower bound program (LLBP) is then given by

$$\min \sum_{i \in V} \sum_{j \in V} \left(c_{ij} + \sum_{k=1}^{K} (t_k - s_k)(r_{ijk} + q_{ik}/2 + q_{jk}/2) \right) x_{ij}$$

$$+ \sum_{k=1}^{K} (s_k L_k - t_k U_k) \tag{11}$$

s.t.
$$\sum_{i \in V} x_{ij} = \sum_{i \in V} x_{ji} \qquad \forall j \in V \, (j \neq 1,n) \tag{12}$$

$$\sum_{j \in V} x_{1j} = 1 \tag{13}$$

$$\sum_{i \in V} x_{in} = 1 \tag{14}$$

$$x_{ij} \in (0,1) \qquad \forall i,j \in V \tag{15}$$

This program is a minimum cost network flow problem—the problem of sending a flow of value one from vertex 1 (Eq. (13)) to vertex n (Eq. (14)) subject to the flow conservation constraints (Eq. (12)) with a capacity limit of one on each arc (Eq. (15)) and the cost of using an arc (i,j) being given by

$$C_{ij} = c_{ij} + \sum_{k=1}^{K} (t_k - s_k)(r_{ijk} + q_{ik}/2 + q_{jk}/2) \tag{16}$$

This minimum cost network flow problem can be easily solved and provides (for any $s_k,t_k \geq 0$) a lower bound on the solution to RCSP.

Note here that if we require the solution to RCSP to be elementary then we can strengthen LLBP by placing a capacity limit of one on each vertex (the capacity of a vertex in the network flow problem representing the number of times that the vertex can be used in the solution to RCSP).

If the original network G is g-elementary then, by definition, the constraints relating to the lower resource limits are redundant and can be neglected. Hence in such cases we modify LLBP by setting $s_k = 0$ $(k = 1, \ldots, K)$ and note that the matrix (C_{ij}) cannot contain any negative cost arcs (or negative cost circuits).

If (C_{ij}) contains no negative cost circuits then LLBP can be regarded as an unconstrained shortest path problem—the problem of finding the shortest elementary path from vertex 1 to vertex n. Computationally this unconstrained shortest path problem can be solved by the algorithm of Dijkstra [15] (requiring $O(n^2)$ operations) if $C_{ij} \geq 0 \; \forall \, (i,j) \in A$ and by the algorithm of Yen [56] (requiring $O(n^3)$ operations) if $C_{ij} < 0$ for some $(i,j) \in A$.

Note here that if, for a particular set of lagrange multipliers, (C_{ij}) contains no negative cost circuits then we were justified in neglecting Eq. (8) in forming LLBP since that equation is automatically satisfied by the solution to LLBP (and similarly for Eq. (10)).

In the next section we discuss the use of subgradient optimization in an attempt to maximize the lower bound that we obtain from LLBP.

IV. THE SUBGRADIENT PROCEDURE

We used subgradient optimization [17,18,25] in an attempt to maximize the lower bound from LLBP. The procedure we adopted was as follows:

(1) Set initial values for the multipliers of $s_k = t_k = 0\ (k = 1, \ldots, K)$. Initialize $Z_{\max}$ (the maximum lower bound found) and B (the iteration count) using $Z_{\max} = -\infty$ and $B = 0$.

(2) Solve LLBP with the current set of multipliers and let the solution be (X_{ij}). The corresponding lower bound Z_{LB} is given by

$$Z_{LB} = \sum_{i \in V} \sum_{j \in V} C_{ij} X_{ij} + \sum_{k=1}^{K} (s_k L_k - t_k U_k) \tag{17}$$

(3) If $Z_{LB} > Z_{\max}$ then set $Z_{\max} = Z_{LB}$.

(4) If (X_{ij}) is a feasible solution to RCSP then update Z_{UB} (the best upper bound corresponding to a feasible solution to RCSP) accordingly.

(5) Stop if $Z_{\max} = Z_{UB}$ (Z_{UB} is the optimal solution).

(6) Calculate the subgradients

$$G_k = L_k - \sum_{i \in V} \sum_{j \in V} (r_{ijk} + q_{ik}/2 + q_{jk}/2) X_{ij} \qquad k = 1, \ldots, K \tag{18}$$

$$H_k = -U_k + \sum_{i \in V} \sum_{j \in V} (r_{ijk} + q_{ik}/2 + q_{jk}/2) X_{ij} \qquad k = 1, \ldots, K \tag{19}$$

(7) Define a step size T by

$$T = f(Z_{UB} - Z_{LB}) \Big/ \left(\sum_{k=1}^{K} ((G_k)^2 + (H_k)^2) \right) \tag{20}$$

where f is a parameter satisfying $0 < f \leqslant 2$ and the denominator of Equation (20) normalises the step size with respect to both sets of subgradients. Update the multipliers by

$$s_k = \max(0,\ s_k + TG_k) \qquad k = 1, \ldots, K \tag{21}$$

$$t_k = \max(0,\ t_k + TH_k) \qquad k = 1, \ldots, K \tag{22}$$

(8) Set $B = B + 1$ and go to step (2) to resolve the problem if $B < B_{\max}$ ($B_{\max}$ the maximum number of iterations allowed).

(9) Resolve LLBP using the set of lagrange multipliers associated with the best lower bound ($Z_{\max}$) found.

Limited computational experience indicated that appropriate values of f and $B_{\max}$ were $f = 0.25$ and $B_{\max} = 10$ and so these values were used whenever the subgradient procedure was applied.

We noted previously that the solution method for LLBP depends upon the nature of the matrix (C_{ij}). It may be that, for ease of solving LLBP, we wish (for example)

to have $C_{ij} \geq 0 \ \forall \ (i,j) \in A$ so that we can use the Dijkstra [15] algorithm requiring only $O(n^2)$ operations to solve LLBP. It is clear that we can attempt to adjust the lagrange multipliers to achieve this (e.g., consider each arc $(i,j) \in A$ in turn and for each arc for which $C_{ij} < 0$ increase some t_k [k such that $r_{ijk} + q_{ik}/2 + q_{jk}/2 > 0$] until $C_{ij} = 0$). Other heuristics (e.g., based on the detection of negative cost circuits and the adjustment of C_{ij} for the arcs in each circuit) can also be used.

At the end of the subgradient procedure we may have found the optimal solution to RCSP but if not we resolve the problem using a tree search procedure. Before considering this tree search procedure, however, we discuss the tests that can be used to eliminate vertices and arcs from the problem.

V. PROBLEM REDUCTION

Various tests can be carried out that will reduce the size of the problem (in terms of vertices/arcs) that we consider. These tests can be categorized into two types: resource based tests and length based tests.

A. Resource Based Tests

Let R_{ijk} be the least amount of resource k that we can use in going from vertex i to vertex j (including any of resource k used at i and j) then any vertex i for which there exists k such that

$$R_{1ik} + R_{ink} - q_{ik} > U_k \tag{23}$$

can be eliminated from the problem as it cannot lie on the optimal path. Similarly, any arc (i,j) for which there exists k such that

$$R_{1ik} + r_{ijk} + R_{jnk} > U_k \tag{24}$$

can also be eliminated from the problem.

Note that for a fixed value of k the values $(R_{1ik} \ \forall \ i \in V)$ and $(R_{ink} \ \forall \ i \in V)$ can be calculated by two applications of the shortest path algorithm of Dijkstra [15] with length matrix (r_{ijk}). In calculating $(R_{ink} \ \forall \ i \in V)$ we reverse the direction of the arcs and calculate the paths from n to all vertices $i \in V$. Similar reductions were first given by Aneja, Aggarwal, and Nair [1].

B. Length Based Tests

Let D_{ij} represent the optimum value of LLBP with the original origin vertex 1 replaced by vertex i and the original destination vertex n replaced by vertex j. If the matrix (C_{ij}) contains no negative cost circuits then D_{ij} is equal to the length of the shortest elementary path from i to j using the lagrangean lengths (C_{ij}) plus the constant term $\sum_{k=1}^{K}(s_k L_k - t_k U_k)$. Then any vertex i which satisfies

$$D_{1i} + D_{in} - \sum_{k=1}^{K}(s_k L_k - t_k U_k) > Z_{UB} \tag{25}$$

can be eliminated from the problem as including it in the solution would force the lower bound above the upper bound. Similarly, any arc (i,j) for which

$$D_{1i} + C_{ij} + D_{jn} - \sum_{k=1}^{K} (s_k L_k - t_k U_k) > Z_{UB} \tag{26}$$

can be eliminated from the problem.

As in the case of the resource based tests (D_{1i}) and (D_{in}) can be calculated by two applications of Dijkstra's algorithm (assuming (C_{ij}) contains no negative cost arcs). Similar expressions to Eqs. (25) and (26) exist which involve the original length matrix (c_{ij}) rather than lagrangean lengths.

VI. THE TREE SEARCH PROCEDURE

As mentioned previously, in the event that the subgradient procedure terminates without having found the optimal solution to RCSP we resolve the problem using a tree search procedure. In the computational results reported later we were concerned with problems where the network G was g-elementary (and consequently the optimal solution to RCSP was elementary). The tree search procedure described below is designed for such problems.

We used a binary depth-first tree search procedure constructing an elementary path from the origin vertex and computing, at each tree node, a lower bound for the optimal completion of this path.

A. Initial Tree Node

(a) We first performed the resource based problem reduction tests.

(b) We then attempted to find a feasible solution to RCSP by simply carrying out the subgradient procedure with an arbitrary initial upper bound Z_{UB} (equal to 50 multiplied by the length of the unconstrained shortest path). In the event that this procedure failed to find a feasible solution the arbitrary upper bound was used; otherwise Z_{UB} was reset to the value of the best feasible solution found.

(c) We then performed the subgradient procedure again. The set of multipliers at the end of the subgradient procedure were not altered from this point on.

(d) The length based problem reduction tests were then performed.

B. Other Tree Nodes

(1) Branch Selection Rule

In forward branching we branched by continuing the construction of the path from the origin (vertex 1) to vertex e (the end of the path formed so far in the tree). We choose the arc (e,j) for which $X_{ej} = 1$ in the optimal solution of LLBP at the current tree node. Note here that we have $e = 1$ in forward branching from the initial tree node.

(2) Problem Reductions

At any tree node we have (by branching) constructed a path from the origin vertex 1 to vertex e. This path has length w (in terms of the original lengths (c_{ij})); length W (in terms of lagrangean lengths (C_{ij})); and resource usage given by the vector (B_k). Then any vertex j ($(e,j) \in A$) for which there exists k such that

$$B_k + r_{ejk} + R_{jnk} > U_k \tag{27}$$

can be eliminated from consideration in all succeeding tree nodes.

Similar expressions to eq. (27) valid for both lagrangean lengths and original lengths (involving W and w, respectively) are also easily derived. All shortest path lengths (R_{jnk} $\forall j \in V, k = 1, \ldots, K$ and D_{jn} $\forall j \in V$) were computed once before branching occurred and were not updated during the tree search.

In addition, in calculating the optimal lower bound completion of the path to e—which involves calculating the shortest path from vertex e to vertex n—we may have calculated the shortest path from e to other vertices (e.g., as will happen if we use the Dijkstra algorithm to solve LLBP). If this set of vertices is given by P then any vertex $j \in P$ for which

$$W + D_{ej} + D_{jn} - \sum_{k=1}^{K} (s_k L_k - t_k U_k) > Z_{UB} \tag{28}$$

can be eliminated from consideration in all succeeding tree nodes.

(3) Backtracking

We can backtrack in the tree when $Z_{LB} > Z_{UB}$ or when no vertex j ($(e,j) \in A$) can be found that has not been eliminated (by the reduction tests) from consideration at the current node in the tree.

VII. COMPUTATIONAL RESULTS

The algorithm was programmed in FORTRAN and run on a CDC 7600 using the FTN compiler with maximum optimisation for a number of randomly generated networks.

For all the problems attempted no lower limits on the resources used by the optimal path were imposed and all arc lengths were nonnegative. This ensured that we were dealing with a g-elementary network and enabled us to use the Dijkstra algorithm to solve LLBP for all values of the lagrange multipliers. For each problem the upper limits on resource usage were calculated by finding the resources used on the unconstrained shortest path and reducing these by a certain percentage. Two types of problems were generated.

The first type (denoted by HZ) uses the scheme outlined by Handler and Zang [23] and makes the amount of resource used by any arc inversely related to the length of the arc. For the second type (denoted by BC) the resources used by each arc, and the length of each arc, were integers uniformly and independently drawn from [0,5]. In an attempt to enforce a reasonable cardinality on the optimal path each arc (i,j) had i randomly generated from $[1,n-1]$ and j from $[i+1,\min(n,i+n/4)]$.

TABLE I. Computational results.

Problem number	Number of vertices (n)	Problem type	Number of arcs ($\mid A \mid$)	Number of resources (K)	Percentage reduction (%)	Shortest path length	Initial tree node: Best lower bound (Z_{max})	Initial tree node: Best upper bound (Z_{UB})	Initial tree node: Sub-gradient time CDC 7600 (seconds)	Initial tree node: After reduction Number of: Vertices	Initial tree node: After reduction Number of: Arcs	Initial tree node: Time CDC 7600 (seconds)	Number of tree nodes	Optimal solution value	Total time CDC 7600 (seconds)
1	100	HZ	955	1	10	80	88.3	142	0.3	9	12	1.8	7	131	1.9
2					20		131	131	0.2	—	—	—	—	131	0.9
3		BC	959		10	1	1.44	2	0.4	13	14	1.8	11	2	1.9
4					20		2	2	0.2	—	—	—	—	2	1.0
5	100	HZ	990	10	10	79	81.9	100	0.9	8	10	4.5	9	100	4.6
6					20		91.4	119	0.9	7	8	4.5	7	100	4.6
7		BC	999		10	3	3.91	9	0.8	83	364	4.3	63	6	4.4
8					20		3.77	150	0.7	89	731	4.3	1069	14	6.3
9	200	HZ	2040	1	10	230	420	420	0.4	—	—	—	—	420	2.0
10					20		420	420	0.4	—	—	—	—	420	2.0
11		BC	1971		10	6	6	6	0.8	—	—	—	—	6	4.0
12					20		6	6	0.8	—	—	—	—	6	3.9
13	200	HZ	2080	10	10	200	448	448	0.9	—	—	—	—	448	5.2
14					20		656	10000	1.6	9	10	9.2	11	—	9.3
15		BC	1960		10	5	6.2	9	1.7	58	102	9.1	93	9	9.2
16					20		5	250	1.4	177	1619	8.9	1293	17	12.1
17	500	HZ	4858	1	10	455	487	690	2.5	9	10	10.5	11	652	10.6
18					20		512	690	2.4	9	10	10.4	11	652	10.5
19		BC	4978		10	6	6	6	2.9	—	—	—	—	6	11.1
20					20		6	6	2.2	—	—	—	—	6	6.4
21	500	HZ	4847	10	10	611	858	858	2.1	—	—	—	—	858	13.6
22					20		858	858	2.1	—	—	—	—	858	13.1
23		BC	4868		10	3	3.34	4	5.1	25	30	26.2	13	4	26.3
24					20		3.74	5	5.1	42	57	26.2	33	5	26.3

Table I shows the results of the algorithm for the problems attempted. The percentage reduction column in that table gives the percentage by which the resources used on the unconstrained shortest path (whose length is given in the following column) were reduced to give the upper resource limits for each problem. Under the columns labeled initial tree node we show the maximum lower bound and the best upper bound found by the subgradient procedure (as well as the time taken by that procedure). The number of vertices and arcs remaining once all the problem reduction tests at the initial tree node had been performed are also shown.

These results show that little difficulty was experienced in solving any of the 24 problems attempted. Nearly half of the problems were solved without any branching being necessary and none of the problems took more than 30 seconds on the CDC 7600.

For problems 8 and 16 the upper bounds shown were not obtained from feasible solutions but were set in an arbitrary manner. In addition these problems did not have many feasible solutions and as a result the number of nodes in the tree search was much higher than for the other problems. For problem 14 no feasible solution existed but this was easily shown in 11 tree nodes as the resource based reduction tests had eliminated most branching possibilities.

VIII. CONCLUSIONS

An algorithm for the resource constrained shortest path problem based upon a lagrangean relaxation of the problem into a minimum cost network flow problem was presented. Incorporating this relaxation, together with some problem reduction tests, into a tree search procedure enabled us to solve problems with upper resource limits involving up to 500 vertices, 5000 arcs and 10 resources in less than 30 seconds on a CDC 7600.

References

[1] Y. P. Aneja, V. Aggarwal, and K. P. K. Nair, Shortest chain subject to side conditions. *Networks* **13** (1983) 295–302.

[2] Y. P. Aneja, and K. P. K. Nair, The constrained shortest path problem. *Nav. Res. Log. Q* **25** (1978) 549–553.

[3] C. P. Bajaj, Some constrained shortest-route problems. *Unternehmensforschung* **15** (1971) 287–301.

[4] J. E. Beasley, The application of mathematical programming and graph theory in distribution problems. PhD thesis, University of London (1979).

[5] R. Bellman and R. Kalaba, On *k*th best policies. *J. SIAM* **8** (1960) 582–588.

[6] N. Christofides, A. Mingozzi, and P. Toth, Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Math. Prog.* **20** (1981) 255–282.

[7] T. S. Chuah, Constrained shortest path. MSc thesis, Department of Management Science, Imperial College, London (1978).

[8] J. C. N. Climaco and E. Q. V. Martins, A bicriterion shortest path algorithm. *EJOR* **11** (1982) 399–404.

[9] K. L. Cooke and E. Halsey, The shortest route through a network with time-dependent internodal transit times. *J. Math. Anal. Appl.* **14** (1966) 493–498.

[10] N. Deo and C. Pang, Shortest-path algorithms: taxonomy and annotation. *Networks* **14** (1984) 257–323.
[11] M. Desrochers and F. Soumis, A generalized permanent labelling algorithm for the shortest path problem with time windows. Publication number 394A, Centre de recherche sur les transports, University of Montreal, Montreal, Quebec H3C 3J7, Canada (1985).
[12] M. Desrochers, An algorithm for the shortest path problem with resource constraints. Publication number 421A, Centre de recherche sur les transports, University of Montreal, Montreal, Quebec H3C 3J7, Canada. To appear in *Networks*.
[13] J. Desrosiers, P. Pelletier, and F. Soumis, Plus court chemin avec contraintes d'horaires. *RAIRO* **17** (1983) 357–377.
[14] J. Desrosiers, F. Soumis, and M. Desrochers, Routing with time windows by column generation. *Networks* **14** (1984) 545–565.
[15] E. W. Dijkstra, A note on two problems in connection with graphs. *Numerische Math.* **1** (1959) 269–271.
[16] S. E. Dreyfus, An appraisal of some shortest-path algorithms. *Opns. Res.* **17** (1969) 395–412.
[17] M. L. Fisher, The lagrangean relaxation method for solving integer programming problems. *Mgmt. Sci.* **27** (1981) 1–18.
[18] M. L. Fisher, An applications orientated guide to lagrangean relaxation. *Interfaces* **15** (1985) 10–21.
[19] R. W. Floyd, Algorithm 97—Shortest path. *Comm. of ACM* **5** (1962) 345.
[20] L. R. Ford, Network flow theory. Report number P-923, The Rand Corporation, Santa Monica, CA (1956).
[21] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, p. 214. W. H. Freeman, San Francisco (1979).
[22] J. Halpern and I. Priess, Shortest path with time constraints on movement and parking. *Networks* **4** (1974) 241–253.
[23] G. Y. Handler and I. Zang, A dual algorithm for the constrained shortest path problem. *Networks* **10** (1980) 293–310.
[24] P. Hansen, Bicriterion path problems. In G. Fandel and T. Gal (Eds.) *Multiple Criteria Decision Making Theory and Application*, (pp. 109–127). Proceedings of a conference held at Hagen/Konigswinter, West Germany, August 20–24, 1979, Springer-Verlag, Berlin (1980).
[25] M. Held, P. Wolfe, and H. P. Crowder, Validation of subgradient optimisation. *Math. Prog.* **6** (1974) 62–88.
[26] M. I. Henig, The shortest path problem with two objective functions. *EJOR* **25** (1985) 281–291.
[27] W. Hoffman, and R. Pavley, A method for the solution of the Nth best path problem. *J. ACM.* **6** (1959) 506–514.
[28] D. J. Houck, J. C. Picard, M. Queyranne, and R. R. Vemuganti, The travelling salesman problem as a constrained shortest path problem: theory and computational experience. *Opsearch* **17** (1980) 93–109.
[29] T. Ibaraki, Algorithms for obtaining shortest paths visiting specified nodes. *SIAM Review* **15** (1973) 309–317.
[30] J. M. Jaffe, Algorithms for finding paths with multiple constraints. *Networks* **14** (1984) 95–116.
[31] P. A. Jensen and R. C. Berry, A constrained shortest path algorithm. Paper presented at the 39th National ORSA Meeting, Dallas, Texas, USA. Abstract given in *ORSA Bulletin* **19** (1971) B-139.
[32] H. C. Joksch, The shortest route problem with constraints. *J. Math. Anal. Appl.* **14** (1966) 191–197.
[33] R. Kalaba, On some communication network problems, combinatorial analysis. *Proc. Symp. Appl. Math.* **10** (1960) 261–280.

[34] N. Katoh, T. Ibaraki, and H. Mine, An efficient algorithm for k shortest simple paths. *Networks* **12** (1982) 411–427.
[35] A. Kershenbaum, W. Hsieh, and B. L. Golden, Constrained routing in large sparse networks. In *Conference record of the IEEE International Conference on Communications*, pp. 38.14–38.18 (1976).
[36] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, p. 92. Holt, Rinehart and Winston, New York (1976).
[37] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, (Eds.) *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, New York (1985).
[38] R. P. Loui, Optimal paths in graphs with stochastic or multidimensional weights. *Comm. of ACM*. **26** (1983) 670–676.
[39] E. Q. V. Martins, On a multicriteria shortest path problem. *EJOR* **16** (1984) 236–245.
[40] M. Minoux and C. Ribeiro, A transformation of hard (equality constrained) knapsack problems into constrained shortest path problems. *Oper. Res. Lett.* **3** (1984) 211–214.
[41] M. Mori and T. Nishimura, Solution of the routing problem through a network by matrix method with auxiliary nodes. *Transportation Res.* **1** (1967) 165–180.
[42] A. Perko, Implementation of algorithms for K shortest loopless paths. *Networks* **16** (1986) 149–160.
[43] A. K. Pujari, S. Agarwal, and V. P. Gulati, A note on the constrained shortest-path problem. *Nav. Res. Log. Q.* **31** (1984) 87–94.
[44] C. Ribeiro and M. Minoux, A heuristic approach to hard constrained shortest path problems. *Discrete Appl. Math.* **10** (1985) 125–137.
[45] C. Ribeiro and M. Minoux, Solving hard constrained shortest path problems by lagrangean relaxation and branch-and-bound algorithms. To appear in *Methods of Operations Research*. Springer-Verlag, New York.
[46] M. Rosseel, Comments on a paper by Romesh Saigal: a constrained shortest route problem. *Opns. Res.* **16** (1968) 1232–1234.
[47] R. Saigal, A constrained shortest route problem. *Opns. Res.* **16** (1968) 205–209.
[48] J. P. Saksena and S. Kumar, The routing problem with 'K' specified nodes. *Opns. Res.* **14** (1966) 909–913.
[49] D. R. Shier, Iterative methods for determining the k shortest paths in a network. *Networks* **6** (1976) 205–229.
[50] D. R. Shier, On algorithms for finding the k shortest paths in a network. *Networks* **9** (1979) 195–214.
[51] P. Vincke, Problèmes multicritères. *Cahiers du Centre d'Etudes de Recherche Opérationnellle* **16** (1974) 425–439.
[52] A. Warburton, Approximation of pareto optima in multiple-objective, shortest-path problems. *Opns. Res.* **35** (1987) 70–79.
[53] D. J. White, The set of efficient solutions for multiple objective shortest path problems. *Comput. Oper. Res.* **9** (1982) 101–107.
[54] D. J. White, *Optimality and Efficiency*, p. 158, Wiley, New York (1982).
[55] C. Witzgall and A. J. Goldman, Most profitable routing before maintenance. Paper presented at the 27th National ORSA Meeting, Boston, Massachusetts, USA. Abstract given in *ORSA Bulletin* **13** (1965) B-82.
[56] J. Y. Yen, An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quart. Appl. Math.* **27** (1970) 526–530.
[57] J. Y. Yen, Finding the k-shortest, loopless paths in a network. *Man. Sci.* **17** (1971) 712–716.

Received February, 1984
Accepted May, 1988