Comput. & Ops Res. Vol. 7, pp. 239–249
Pergamon Press Ltd., 1980. Printed in Great Britain

DEGREE-CONSTRAINED MINIMUM SPANNING TREE

SUBHASH C. NARULA
Rensselaer Polytechnic Institute, Troy, NY 12181, U.S.A.

and

CESAR A. HO
Universidad Nacional Technica de Piora, Peru

Scope and purpose—The problem of generating a minimum spanning tree, when the number of arcs incident to any node are specified, is posed. Three procedures to construct such a tree are proposed. These procedures are compared in terms of computer time required to solve a problem. Some guidelines are proposed to assist in the selection of an algorithm for a given problem.

Abstract—In this paper the problem of a degree-constrained minimum spanning tree (DCMST) is defined. The problem is formulated as a linear 0–1 integer programming problem. A primal and a dual heuristic (construction) procedure and a branch-and-bound algorithm are proposed to construct a DCMST. These procedures are illustrated with a simple example. Some computational experience with these algorithms is also reported.

1. INTRODUCTION

In the design of electrical circuits, the following problem is often considered: connect n terminals with the minimum amount of wire. The length of a minimum spanning tree (MST) connecting the terminals gives the required length of wire. However, sometimes the size of the terminals is such that the number of wires incident to a terminal i cannot be more than b_i, $i = 1, \ldots, n$. The solution to this problem may not be the length of a MST. The solution to such problems will be the length of a degree-constrained minimum spanning tree (DCMST).

The problem of finding a DCMST also arises in many other areas such as transportation, communication, plumbing, sewage, etc. The DCMST problem subsumes a number of well known and interesting problems. For example, if $b_i = n - 1$, $i = 1, \ldots, n$, the degree-constraints become redundant and the problem reduces to the MST problem; if the degree at each node is strictly equal to 2, the problem reduces to the travelling salesman problem (TSP). Further, in the planning of some telecommunication systems, the objective is to set up transmission lines to connect users in different places to a common computer center. The number of direct lines to the computer center is exactly equal to the number of users is obviously a special case of the DCMST problem; the problem is known as the order-constrained minimum spanning tree.

The problem and the procedures to construct a MST are well discussed in the literature, (see Dijkstra[1], Kruskal[2] and Prim[3]). Recently, an efficient procedure for solving the travelling salesman problem was proposed by Held and Karp[4, 5] and further improvements suggested by Hansen and Krarup[6]. For the order-constrained minimum spanning tree, Glover and Klingman[7] give an admissible edge exchange procedure. However, the DCMST problem is only mentioned in passing by Obruca[8].

In this paper, we formulate the DCMST problem and propose three procedures to generate such a tree. We illustrate the procedures with a simple example and present some computational experience with the proposed algorithms.

2. PROBLEM FORMULATION

The degree-constrained minimum spanning tree can be stated as follows.

Given a non-directional complete graph $G(V, E)$, with cost (length, time) c_{ij} associated with the edge e_{ij} for every $e_{ij} \in E$, construct a minimum cost spanning tree such that the degree d_i at a node i for every $i \in V$, is less than or equal to b_i.

The problem can be formulated as a linear 0–1 integer programming problem. Let $X_{ij} = 1$ if

an edge $e_{ij} \in E$ is included in the tree and 0 otherwise. Then the problem is

minimize

$$\sum_{\substack{i,j \in V \\ i \neq j}} c_{ij} X_{ij}.$$

subject to

$$\sum_{\substack{j \in V \\ i \neq j}} X_{ij} \leq b_i \qquad \forall i \in V$$

$$\sum_{\substack{j \in V \\ i \neq j}} X_{ij} \geq 1 \qquad \forall i \in V$$

$$\sum_{i,j \in N} X_{ij} \leq |N| - 1 \qquad \forall N \subset V$$

$$X_{ij} = 0 \text{ or } 1 \qquad i, j \in V.$$

3. SOLUTION PROCEDURES

Clearly, the problem can be solved using a linear 0–1 integer programming algorithm. However, for n terminals, the problem has $n(n-1)/2$ variables and $2^n + n - 1$ constraints. Even for a moderate value of n, the problem can become very large very fast, making the linear 0–1 integer programming algorithms unattractive.

We propose two construction procedures and a branch-and-bound algorithm to generate a DCMST.

3.1. *A primal method*

The method starts with a feasible degree-constrained spanning tree (DCST) and moves towards optimality maintaining feasibility at each step.

The construction procedure to determine a starting feasible DCST is a modification of the Prim's algorithm [3] for the MST problem. Prim [3] gives two construction principles to generate a MST. The first principle P1 states that an isolated node i can be connected to the "nearest neighbor" j of i and the second principle P2 states that any "fragment" $V_S \subset V$ can be connected to the "nearest neighbor" j of the fragment. Repeated use of the two principles generates a MST.

To generate a feasible DCST, we modify the second princeple P2′ as follows: any "fragment" $V_S \subset V$ can be connected to the "nearest neighbor" by the shortest edge whose inclusion will not violate any degree constraint. Using P1 and then repeatedly using P2′ will generate a DCST.

The modified Prim's algorithm to generate a feasible DCST can be summarized as follows:

Let the component of an entry i of the F table (Prim[3]) be: $F1(i)$—a node not in the fragment, $F3(i)$—a node in the fragment, and $F2(i)$—the length of the edge joining the node in $F1(i)$ and $F3(i)$.

Step 1. Initialize F table by setting $F3(i) = p$ (any initial node) and $F1(i) = q$ for $i = 1, \ldots, n-1$ and $q = 1, \ldots, n$, $q \neq p$. Set $F(2)(i) =$ cost of the edge e_{pq}, $d_k = 0$ for $k = 1, \ldots, n$ and $m = 1$. Go to Step 2.

Step 2. Let $c^* = F2(i^*) = \min_i F2(i)$, $r = F3(i^*)$ (note $r = p$ for $m = 1$), and $s = F1(i^*)$. If $d_r < b_r$, set $E(m) = e_{rs}$, $d_r = d_r + 1$ and $d_s = d_s + 1$; go to Step 4. If $d_r = b_r$, go to Step 3.

Step 3. For all entries j for which $F3(j) = r$, let $t = F1(j)$ and set $F3(j) = u$, where node u is in the fragment such that $0 < d_u < b_u$ and edge e_{tu} is the shortest edge among the eligible edges joining t to the fragment. Further, set $F2(j) =$ cost of the edge e_{tu}. Go to Step 2.

Step 4. Delete entry i^* from the F table. Compare every $F2(i)$ with $G(i)$—the cost of edge joining a node s to $F1(i)$. If $G(i) < F2(i)$, set $F2(i) = G(i)$ and $F3(i) = s$. Set $m = m + 1$. If $m > n - 1$, stop; otherwise go to Step 2.

The DCST so generated need not be a DCMST. To find a DCST with a lower cost we check each edge e_{ij} in the current tree to see if (i) it can be replaced by a shorter edge e_{vw} or (ii) it can

be replaced by an equal cost edge e_{vw} such that nodes v or w or both do not reach the maximum degree allowed and nodes i or j or both already have the maximum degree.

The improvement edge exchange algorithm can be stated as:

Step 1. Set $m = 1$.

Step 2. Identify the subtrees T_i and T_j created by the deletion of $e_{ij} = E(m)$ from the tree.

Step 3. Find an edge e_{vw}, $v \in T_i$ and $w \in T_j$ such that cost $(e_{vw}) <$ cost (e_{ij}) and $d_v + 1 \leq b_v$, $d_w + 1 \leq b_w$. If a e_{vw} exists, go to Step 6; otherwise, go to Step 4.

Step 4. Check if $d_i = b_i$ or $d_j = b_j$. If true, go to Step 5; otherwise, go to Step 7.

Step 5. Find an edge e_{vw}, $v \in T_i$ and $w \in T_j$ such that cost $(e_{vw}) =$ cost (e_{ij}) and $d_v + 1 \leq b_v$ and $d_w + 1 \leq b_w$. If a e_{vw} exists, go to Step 6; otherwise, go to Step 7.

Step 6. Exchange e_{ij} and e_{vw}. Go to Step 7.

Step 7. Set $m = m + 1$. If $m \leq n - 1$, go to Step 2; otherwise, stop.

3.2 *A "dual" method*

With the MST as a starting solution, the dual method moves towards feasibility and thus optimality by making "admissible" edge exchanges. The procedure is based on the following observations.

First, the cost of a MST provides a lower bound on the cost of a DCMST. Second, it is reasonable to assume that many edges appearing in the MST will also appear in a DCMST. In all probability, the edges common to the MST and the DCMST will be edges incident to a node i with $d_i \leq b_i$. Thus, we can manipulate the edges of a MST to generate a degree-constrained spanning tree. Similar observations were made by Obruca[8] between the MST and the solution to the travelling salesman problem.

The edge exchange procedure, used for manipulating the edges of a MST, is similar to the one by Glover and Klingman[7] and proceeds as follows: Consider a node i such that $d_i > b_i$. For each edge e_{ij} in the tree, find an admissible replacement edge e_{rs} such that (i) the deletion of e_{ij} and the inclusion of edge e_{rs} will result in a tree, (ii) the degree constraint at nodes r or s or both will not be violated, and (iii) the penalty (= cost (e_{rs}) − cost (e_{ij})) is the least among all such edges e_{rs}. We make an exchange between the pair (e_{ij}, e_{rs}) for which the penalty is the least. The exchanges are made until $d_i \leq b_i$ for all i. The procedure can be described as follows:

Step 1. Generate a MST.

Step 2. Search a node i such that $d_i > b_i$. If one such node exists, go to Step 3; otherwise, stop.

Step 3. Let V_i denote the set of nodes incident to node i in the current tree. Compute $p(j) =$ cost (e_{ij}) − cost $(e_{rs}(j))$ for $j \in V_i$, where $e_{rs}(j)$ is the smallest replacement edge that joins the subtrees created by the deletion of the edge e_{ij} such that $d_r + 1 \leq b_r$, $(d_s + 1 \leq b_s)$ if r (or $s) \neq j$ and $d_r \leq b_r$, $(d_s \leq b_s)$ if r (or s) $= j$. Go to Step 4.

Step 4. Let $p(j^*) = \min_j p(j)$. Make the edge exchange for the pair $(e_{ij^*}, e_{rs}(j^*))$. Set $d_i = d_i - 1$, $d_{j^*} = d_{j^*} - 1$, $d_r = d_r + 1$, and $d_s = d_s + 1$. If $d_i \leq b_i$, go to Step 2; otherwise, go to Step 3.

Since the tree so found may not be the DCMST, improvement edge exchanges as described in Section 3.1 are made.

3.3 *A branch and bound procedure*

A branch-and-bound procedure may be used to generate a DCMST. The cost of a MST provides a lower bound and the cost of any feasible DCST can be used as an upper bound on the cost of the DCMST. The branching procedure described here is an adaptation of the method due to Held and Karp[4, 5] for the travelling salesman problem. The branching procedure may be described as follows:

Let X be the set of edges currently included in the tree, and Y be the set of edges currently excluded. At the beginning of the algorithm, the sets X and Y are empty. At any iteration, if the lower bound for a subset (X, Y) is less than the current upper bound; sort the edges, not yet included or excluded, in a decreasing order of the amount by which the lower bound will increase if the edge were excluded. Let t be the number of edges not yet included or excluded and $e(1), \ldots, e(t)$ be the sorted sequence of edges. The subset is then partitioned and new

subsets created as follows:

$$\begin{aligned} X_1 &= X & Y_1 &= Y \cup \{e(1)\} \\ X_2 &= X \cup \{e(1)\} & Y_2 &= Y \cup \{e(2)\} \\ X_2 &= X \cup \{e(1), e(2)\} & Y_3 &= Y \cup \{e(3)\} \\ &\vdots & &\vdots \\ X_k &= X \cup \{e(1), \ldots, e(k-1)\} & Y_k &= Y \cup E_p \end{aligned}$$

where k is the smallest index such that some node p has b_p edges incident to it, and E_p is the set of all edges not yet included or excluded that are incident to the node p. For a subset (X_m, Y_m), a lower bound is calculated as the total length of the MST that includes the edges in X_m and excludes those in Y_m.

In this procedure, the number of nodes of a search tree can become large, requiring excessive core storage space. To reduce the core requirement, the procedure may be modified as follows: always branch to the right and at each node m, keep a "right" bound (the current lower bound) and a "left" bound (the lower bound of the subset (X_{m-1}, Y_{m-1})).

The algorithm may be summarized as follows: Let $R(i)$ and $L(i)$ be the right and the left bounds, respectively, of a subset at level i of the search tree. Further, let $k(i)$ be the subscript of the last entry (X_k, Y_k) generated at level i, as described in the branching procedure.

Step 1. Generate a degree-constrained spanning tree. Set upper bound $\bar{Z}$ = total cost and $i = 1$.

Step 2. Generate a MST. Set $R(i)$ = total cost, $L(i) = \infty$ and $i = i + 1$.

Step 3. Determine subset $(X_{k(i)}, Y_{k(i)})$. Set $R(i)$ = bound of $(X_{k(i)}, Y_{k(i)})$ and $m(i) = k(i) - 1$. Determine set $(X_{m(i)}, Y_{m(i)})$ and let $L(i)$ = bound of $(X_{m(i)}, Y_{m(i)})$. Go to Step 4.

Step 4. If $R(i) < \bar{Z}$, go to Step 5; otherwise, go to Step 6.

Step 5. If the tree corresponding to $R(i)$ is feasible, set $\bar{Z} = R(i)$ and go to Step 6; otherwise, set $i = i + 1$ and go to Step 3.

Step 6. If $L(i) < \bar{Z}$, set $R(i) = L(i)$ and go to Step 7; otherwise, go to Step 8.

Step 7. Set $m(i) = m(i) - 1$. If $m(i) > 0$, determine subset $(X_{m(i)}, Y_{m(i)})$ and set $L(i)$ = bound of $(X_{m(i)}, Y_{m(i)})$; otherwise, set $L(i) = \infty$. Go to Step 4.

Step 8. Set $i = i - 1$. If $i > 1$, go to Step 6; otherwise, stop.

4. AN EXAMPLE

We illustrate the proposed procedures by solving a problem with 9 nodes and restricting the maximum degree to less than or equal to 3.

Consider the graph whose distance matrix (upper triangle) appears in Table 1.

4.1 *The primal method*

The primal method starts by generating a feasible DCST using the modified Prim's algorithm. The F-tables that are generated are given in Table 2. The entries in the second row

Table 1. Distance matrix for the example

Node	1	2	3	4	5	6	7	8	9
1	-	2.24	2.24	3.61	6.71	3.00	5.39	8.00	9.43
2		-	2.00	2.00	4.47	2.83	4.00	7.28	7.62
3			-	4.00	5.66	4.47	6.00	9.22	9.49
4				-	4.00	2.00	2.00	5.39	5.83
5					-	6.00	4.47	7.81	5.10
6						-	2.83	5.00	7.07
7							-	3.61	4.24
8								-	5.00

Table 2. F-Tables for modified Prim's algorithm

		↓							
	F1	2	3	4	5	6	7	8	9
Stage 1	F2	2.24	2.24	3.61	6.71	3.00	5.39	8.00	9.43
	F3	1	1	1	1	1	1	1	1
		↓							
	F1	3	4	5	6	7	8	9	
Stage 2	F2	2.00	2.00	4.47	2.83	4.00	7.28	7.62	
	F3	2	2	2	2	2	2	2	
		↓							
	F1	4	5	6	7	8	9		
Stage 3	F2	2.00	4.47	2.83	4.00	7.28	7.62		
	F3	2	2	2	2	2	2		
			↓						
	F1	5	6	7	8	9			
Stage 4	F2	4.00	2.00	2.00	5.39	5.83			
	F3	4	4	4	4	4			
			↓						
	F1	5	7	8	9				
Stage 5	F2	4.00	2.00	5.00	5.83				
	F3	4	4	6	4				
			↓						
	F1	5	8	9					
Stage 6	F2	4.00	3.61	4.24					
	F3	4	7	7					
	F1	5	9						
Stage 7	F2	4.00	4.24						
	F3	4	7						
			↓						
	F1	5	9						
Stage 8	F2	4.47	4.24						
	F3	7	7						
		↓							
	F1	5							
Stage 9	F2	5.10							
	F3	9							

of the successive F-tables are the distances from the connected fragment to the unconnected terminals at each stage of fragment growth. The entries in the third row of these tables indicate the nearest neighbor in the fragment of the external terminal in question.

The computations are started by entering the first row of the distance table into the F-table. Stage 1 contains the first row of the distance matrix. The shortest entry in the first stage is the first entry (corresponding to edge 1–2); edge 1–2 is selected to enter the fragment. The F-table is then updated to include all the entries corresponding to the shortest edges joining the fragment with the isolated nodes (the isolated nodes at this stage are 3–9). At stage 2, edge 2–3 is selected and this entry is deleted from the table. Stage 3 table remains unchanged as the edges joining the new selected node (node 3) with the remaining isolated nodes are larger than the ones already in the table. The procedure is repeated until F-table of stage 7 is obtained. The shortest entry at this stage corresponds to edge 4–5. However, node 4 already has a degree equal to 3. So, stage 7 F-table is changed replacing entry 4–5 by entry 5–7 which is the shortest edge joining node 5 to the fragment without violating a degree constraint. Similar change is made in the F-table at stage 8 where edge 5–7 is replaced by edge 5–9. The resulting tree is shown in Fig. 1(a).

The edges in the first degree-constrained spanning tree are then inspected to see if any of these can be replaced by a shorter or an equal length edge as described in Section 2.1. Let edge 1–2 be the first edge to be inspected. This creates two subtrees T_1 and T_2 as shown in Fig. 1(b). An equal length edge rejoining T_1 and T_2, without violating a degree constraint is edge 1–3. No

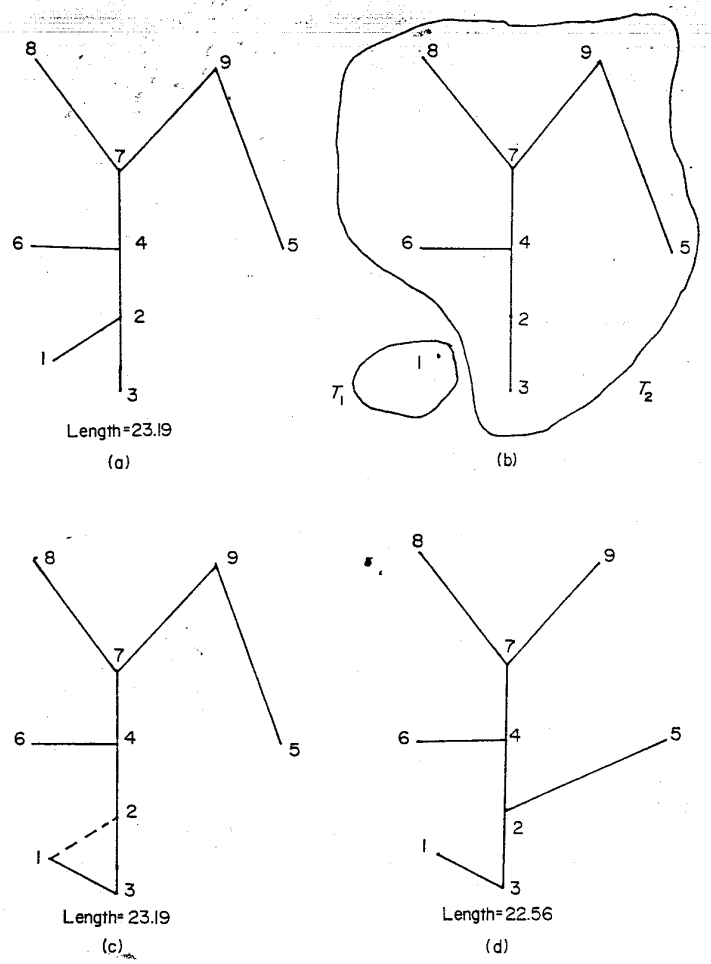

Fig. 1. Trees at various steps of the primal method.

shorter edge is available. Since it only increases the degree of node 3–2 (less than the maximum permissible), we make the exchange. The resulting tree is shown in Fig. 1(c). The rest of the edges in the tree are inspected in a similar manner. Only one more improving edge exchange is possible; edge 5–9 for edge 2–5. Since no more improvement edge exchanges can be made to the tree, the procedure stops. The final tree is given in Fig. 1(d).

4.2 *The dual method*

The dual method starts by generating a MST, using Prim's algorithm. The F-tables are given in Table 3. The procedure followed is similar to the one explained in the primal method to generate the first feasible DCST, with an exception that the F-table at a stage is not changed if a node in the fragment reaches the maximum degree specified. The resulting MST is given in Fig. 2(a).

Note that node 4 has a degree greater than the maximum allowed. One of the edges incident to node 4 must be deleted and replaced by another (not violating a constraint) in such a way that the increase in the total length of the tree is the least. Consider edge 4–7; if it is deleted the MST is divided into two subtrees T_4 and T_7 as in Fig. 2(b). The four shortest edges (different from edge 4–7) rejoining T_4 and T_7, without violating a degree constraints, are edges 6–7, 5–7, 6–8, and 5–9; the shortest of these being edge 6–7. The penalty associated with the pair 4–7 and 6–7 is length (6–7) − length (4–7) = 0.83. Similar analysis is made for all edges incident to node 4 and the pair of edges having the smallest penalty are exchanged. The best exchange is to

Table 3. *F*-Tables for Prim's algorithm

		↓							
	F1	2	3	4	5	6	7	8	9
Stage 1	F2	2.24	2.24	3.61	6.71	3.00	5.39	8.00	9.43
	F3	1	1	1	1	1	1	1	1
		↓							
	F1	3	4	5	6	7	8	9	
Stage 2	F2	2.00	2.00	4.47	2.63	4.00	7.28	7.62	
	F3	2	2	2	2	2	2	2	
		↓							
	F1	4	5	6	7	8	9		
Stage 3	F2	2.00	4.47	2.83	4.00	7.28	7.62		
	F3	2	2	2	2	2	2		
			↓						
	F1	5	6	7	8	9			
Stage 4	F2	4.00	2.00	2.00	5.39	5.83			
	F3	4	4	4	4	4			
			↓						
	F1	5	7	8	9				
Stage 5	F2	4.00	2.00	5.00	5.83				
	F3	4	6	4					
			↓						
	F1	5	8	9					
Stage 6	F2	4.00	3.61	4.24					
	F3	4	7	7					
		↓							
	F1	5	9						
Stage 7	F2	4.00	4.24						
	F3	4	7						
		↓							
	F1	9							
Stage 8	F2	4.24							
	F3	7							

replace edge 4–7 by edge 6–7. The length of the resulting tree is the length of the MST plus the penalty. The resulting tree is in Fig. 2(c).

The improving edge exchanges are next attempted in the manner described for the primal method. Only one exchange is possible, edge 1–3 for edge 1–2. The final tree is given in Fig. 2(d).

4.3 *The branch and bound method*

The branch and bound method starts with a feasible DCST as in the primal method (Fig. 1(a)), and its length as the upper bound. Thus, the first upper bound on the length is $\bar{Z} = 23.19$. The MST obtained at the beginning of the dual method (Fig. 2(a)) provides an initial lower "right" bound $\underline{ZR}_0 = 22.09$. The initial lower "left" bound is $\underline{ZL}_0 = \infty$.

In the beginning, the set NXY is the set of all edges in the initial MST. Thus $NXY_0 = \{1\text{–}2, 2\text{–}3, 2\text{–}4, 4\text{–}5, 4\text{–}6, 4\text{–}7, 7\text{–}8, 7\text{–}9\}$. The set X_0 of included edges and the set Y_0 of excluded edges are initially empty. The penalty for each edge in set NXY_0 is calculated. (The penalty of an edge i–j is the difference between the length of edge i–j and the length of the shortest edge rejoining the two subtrees, T_i and T_j, created by the deletion of edge i–j.) The penalties are calculated as in the dual method with the difference that here the degree constraints are not considered. After the penalties are calculated the edges are arranged in order of decreasing penalties. The computations for this step appear in Table 4.

Branching starts by passing edges from the set NXY_0 to a set X_1 of edges included as follows: $X_1 = \{7\text{–}8, 4\text{–}7, 4\text{–}6, 2\text{–}4\}$. The transfer of edges stops when edge 2–4 is included because at this point there are 3 edges incident to node 4. The set Y_1 includes the edges (not already in a previous set X or Y) incident to node 4. Thus, $Y_1 = \{1\text{–}4, 3\text{–}4, 4\text{–}5, 4\text{–}8, 4\text{–}9\}$. At this

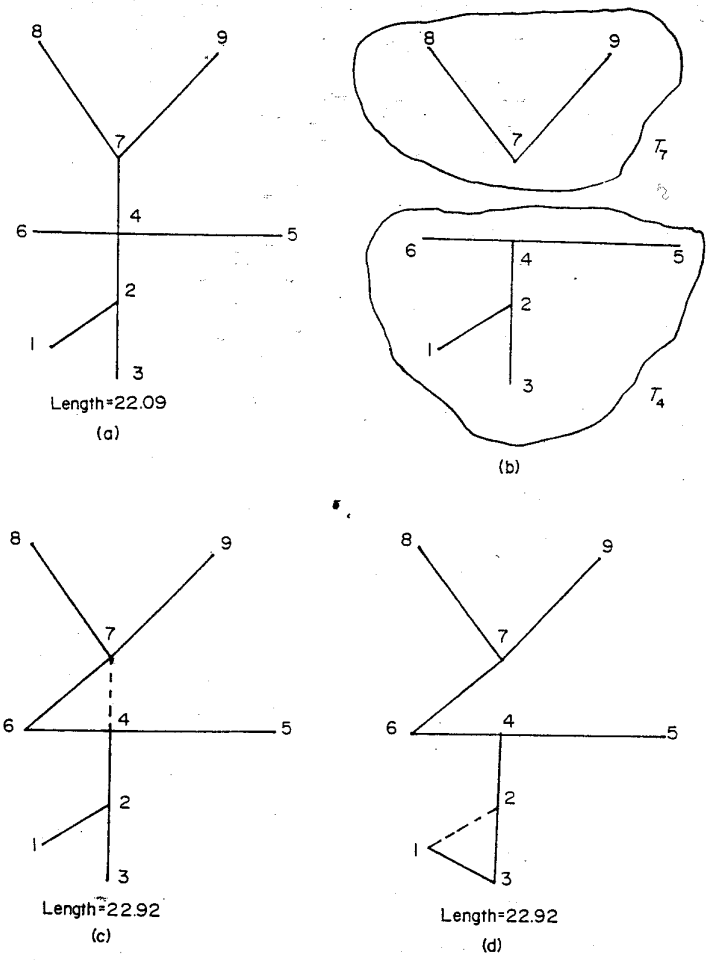

Fig. 2. Trees at various steps of the dual method.

Table 4. Table of penalty functions

Edge	Penalty	Replacement Edge
7-8	1.39	6-8(or 8-9)
4-7	0.83	6-7
4-6	0.83	2-6(or 6-7)
2-4	0.83	2-6
7-9	0.76	8-9
4-5	0.47	2-5(or 5-7)
2-3	0.24	1-3
1-2	0.00	1-3

stage, a tree of minimum length, that includes the edges in X_1 and excludes those in Y_1, is generated. This tree appears in Fig. 3(a).

The right bound is not $\underline{ZR}_1 = 22.56$. The left bound is $\underline{ZL}_1 = \underline{ZR}_0 +$ penalty of the last edge in X_1 (edge 2–4), i.e. $\underline{ZL}_1 = 22.09 + 0.83 = 22.92$. At this stage, $\underline{ZR}_1$ is smaller than the current upper bound $\bar{Z}$ and the tree is infeasible (degree of edge 2 is equal to 4). The computations are repeated with set $NXY_1 = \{1\text{–}2, 2\text{–}3, 2\text{–}5, 7\text{–}9\}$, a set of edges not yet included or excluded, and

sets X_1 and Y_1. The penalties are calculated as before and the bounds at this stage are:

$$X_2 = X_1 \cup \{7\text{–}9\}$$
$$Y_2 = Y_1 \cup \{1\text{–}7, 2\text{–}7, 3\text{–}7, 5\text{–}7, 6\text{–}7\}$$
$$\underline{ZR}_2 = 22.56 \text{ infeasible}$$
$$\underline{ZL}_2 = 23.32$$
$$N\overline{XY}_2 = \{2\text{–}5, 2\text{–}3, 1\text{–}2\}.$$

Further branching continues as follows:

$$X_3 = X_2 \cup \{2\text{–}5, 2\text{–}3\}$$
$$Y_3 = Y_2 \cup \{1\text{–}2, 2\text{–}6, 2\text{–}7, 2\text{–}8, 2\text{–}9\}$$
$$\underline{ZR}_3 = 22.56.$$

Since at this stage, a feasible DCST is obtained, $\underline{ZR}_3$ becomes a new upper bound, i.e. $\bar{Z} = \underline{ZR}_3$. The left bounds ($\underline{ZL}_2$, $\underline{ZL}_1$, $\underline{ZL}_0$) are inspected, but none of these is smaller than the new upper bound $\bar{Z}$. The algorithm stops. The final tree is shown in Fig. 3(b).

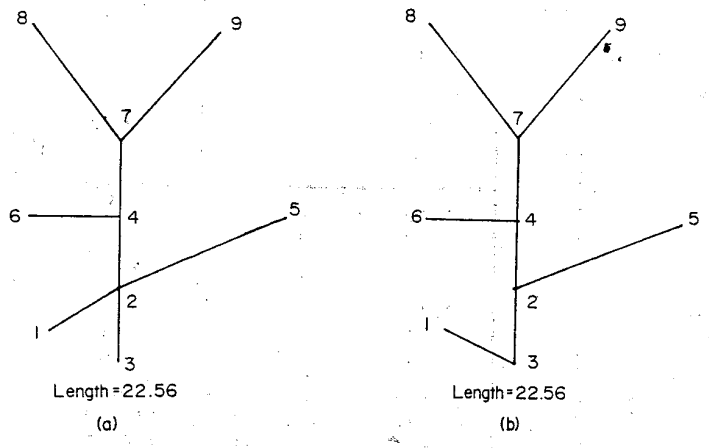

Fig. 3. Trees generated by the branch and bound method.

5. COMPUTIONAL EXPERIENCE

At this point it may be observed that the branch and bound procedure (if all the branches are examined implicitly) will converge to a global optimal solution. The other procedures may not give a global optimal solution, as seen in the example (Section 3). This happens since only one (improving) edge exchange is considered at a time which may lead us to a local optimum. By modifying the method of improving edge exchanges it is possible to make all procedures to converge to a global optimal solution.

The three algorithms were programmed in FORTRAN IV for a CDC CYBER 173. The three-label-per-node labelling procedure of Scions[9] as modified by Johnson[10] was used to store a tree. The procedure of Glover *et al.*[11] was used for updating the tree. To find the admissible edge exchanges in the dual method, the labelling method of Glover and Klingman[7] was used. Further, in the dual and the branch-and-bound algorithms, the MST was found using Prim's algorithm.

The algorithms were tested on a number of randomly generated problems with $n = 30$, 50 and 100. The test problems were created by generating the coordinates of the nodes at random; and using the Euclidean distance between nodes i and j, $(i, j = 1, \ldots, n)$ as c_{ij}. It can be easily shown that when Euclidean distance between nodes is used for c_{ij}, the maximum degree at any node of a MST is no more than 6. Further, the maximum degree at node i was specified to be less than or equal to $b_i = b$ for $i = 1, \ldots, n$. The values of b were restricted to 2, 3 and 4 as the maximum degree of the MST's for the generated problem was no more than 5.

The DCMST with $b = 2$ can be obtained by deleting the largest edge from the solution of the travelling-salesman problem. Since efficient algorithms are available for solving a TSP (see Hansen and Krarup[6]), results for $b = 2$ are not reported here.

Table 5 gives the average solution times, the standard deviation and the number of problems for which the optimal solution was found. Note that the statistics for $n = 30$ and $n = 50$ are based on 50 problems, and for $n = 100$ are based on 20 problems.

For problems for which the methods did not find an optimal solution, Table 6 gives the maximum percentage by which the solution obtained differed from the optimal solution (or the cost of the MST if the optimal solution was not known).

Table 5. Comparative results: Average Solution Time(CPU Secs); (Standard deviation); Number of optimal solutions found

n	b	Method		
		Primal	Dual	Branch-and-Bound
30	3	1.342 (0.042) 46	1.188 (0.138) 50	0.711 (0.876) 50
	4	1.352 (0.190) 50	1.127 (0.109) 50	0.250 (0.252) 50
50	3	9.049 (0.256) 44	8.697 (0.426) 50	4.844 (8.324) 50
	4	9.034 (0.182) 50	8.578 (0.562) 50	0.961 (1.116) 50
100	3	139.828 (2.630) 3	137.482 (2.733) 6	212.923 (56.271) 6
	4	135.542 (3.332) 18	136.132 (5.384) 19	20.068 (5.175) 20

Table 6. Maximum percent difference between obtained and optimal solution

n	b	Method		
		Primal	Dual	Branch-and-Bound
30	3	1.1	0.0	0.0
	4	0.0	0.0	0.0
50	3	0.8	0.0	0.0
	4	0.0	0.0	0.0
100	3	1.6*	1.6*	0.37*
	4	1.2	1.2	0.0

*Compared with the MST.

Table 7. Methods in order of preference

b \ n	30	50	100
3	1. Branch-and-Bound 2. Dual Primal	1. Branch-and-Bound 2. Dual Primal	1. Dual 2. Primal 3. Branch-and-Bound
4	1. Branch-and-Bound 2. Dual Primal	1. Branch-and-Bound 2. Dual Primal	1. Branch-and-Bound 2. Primal Dual

6. DISCUSSION AND RECOMMENDATIONS

Before making the final recommendations, it is important to observe the following.

Although more efficient for $n = 30$ and 50 the branch-and-bound procedure requires more core storage space (approx. twice) compared to the primal and the dual methods. Also, the solution times increase rapidly as the number of nodes n is increased and/or b is decreased.

The primal method generally generates a better initial feasible solution in less time than the initial feasible solution by the dual method. Further, the primal method maintains feasibility at each step.

Based on the preceding observations and limited computational experience (Tables 5 and 6), Table 7 lists the methods in terms of the average time required to solve a problem of given size and thus provides some guidance to the reader in selecting an algorithm.

Acknowledgements—The authors wish to thank Charles G. DeWald for suggesting problem. The computer time for the project was provided by the Department of Industrial Engineering, State University of New York at Buffalo.

REFERENCES

1. E. W. Dijkstra, A note on two problems in connexion with graphs. *Numerische Mathematik* **1**, 269–271 (1959).
2. J. Kruskal, On the shortest spanning subtree of a graph and the travelling-salesman problem. *Proc. Am. Math. Soc.* **7**, 48–50 (1956).
3. R. C. Prim, Shortest connection networks and some generalizations. *Bell Systems Tech. J.* **36**, 1389–1401 (1957).
4. M. Held and R. M. Karp, The travelling-salesman problem and minimum spanning trees. *Ops Res.* **18**, 1138–1162 (1970).
5. M. Held and R. M. Karp, The travelling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming* **1**, 6–25 (1971).
6. K. H. Hansen and J. Krarup, Improvements of the Held–Karp algorithm for the symmetric travelling-salesman problem. *Mathematical Programming* **7**, 87–96 (1974).
7. F. D. Glover and D. Klingman, Finding minimum spanning trees with a fixed number of links at a node. *Res. Rep.* CS 169, Center for Cybernetics Studies, The University of Texas at Austin (1974).
8. A. K. Obruca, Spanning tree manipulation and the travelling-salesman problem. *Comp. J.* **10**, 374–377 (1968).
9. H. Scions, The compact representation of a rooted tree and the transportation problem. *Int. Symp. Math. Programming*, London (1964).
10. E. Johnson, Networks and basic solutions. *Ops Res.* **14**, 619–623 (1966).
11. F. D. Glover, D. Karney and D. Klingman, The augmented predecessor index method for locating stepping stone paths and assigning dual prices in distribution problems. *Transpn Sci.* **6**, 171–180 (1972).