Pergamon

0360-8352(95)00036-4

Computers ind. Engng Vol. 28, No. 4, pp. 689–700, 1995

0360-8352/95 $9.50 + 0.00

A TABU SEARCH FOR THE SURVIVABLE FIBER OPTIC COMMUNICATION NETWORK DESIGN

SEOK J. KOH and CHAE Y. LEE

Department of Management Science, Korea Advanced Institute of Science and Technology, 373-1 Kusung-Dong, Taejon, Korea

(Received June 1995)

Abstract—A tabu search procedure is developed to solve fiber optic communication network design problems with survivability constraints. Two systematic improving heuristics: delete–add and delete–link procedures are presented. The conditions for the candidate links to be added and deleted in the two procedures are examined by considering the feasible structures of the survivable network. A local improvement procedure is considered by combining the two heuristics for the downhill move in the search procedure. Computational results show that the proposed tabu search outperforms the best known heuristic procedure in the literature.

1. INTRODUCTION

The fiber optic technology is one of the major components in the modern communication networks. The transmission medium is cost effective and reliable. It also provides nearly unlimited capacity. The advent of the fiber optic technology has brought a significant shift in network design problems. The high capacity of fiber facilities enables each link to carry larger amounts of traffic than the traditional copper-based technology. This situation, however, increases the possibility of severe service loss in the event of link or node failure. Therefore, in order to provide reliable service to customers, it is desirable to enhance network survivability at the expense of increased cost. As a result, a new objective becomes a cost-effective network design with a guaranteed level of survivability.

Recent studies by several researchers have focused on the design of survivable fiber communication networks. Network design models with connectivity constraints [1, 2, 5, 6, 8, 9] are provided. In the study, the connectivity between the origin and destination pair is taken as a measure for survivability from the link or node failure. Two-connected topology is suggested for the survivability in the fiber optic communication networks.

Bienstock *et al.* [1] and Monma *et al.* [8] have developed several properties on the structure of the optimal two-connected network. Groetschel and Monma [5] have also provided a preliminary study of the problem from a polyhedral point of view. Based on the theoretical study, Monma and Shallcross [9] and Cardwell *et al.* [2] have proposed diverse protective routing techniques and generated very effective solutions for the two connectivity problem. Groetschel *et al.* [6] have employed a cutting plane method using the polyhedral properties of network to obtain exact solutions.

The network design problem considered in this paper consists of switching offices each of which is either special or regular, and potential links between offices where fiber facilities could be installed. Each link in the communication network has an associated nonnegative cost of installing the fiber facilities. For the survivability of the network, the network configuration requires at least two node disjoint paths between each pair of special offices. Such a requirement, however, is not necessary for the regular offices. The topology provides protection against any single node or link failure. Figure 1 illustrates the survivability of the network from any single node or link failure. Any failure of a special office can be protected by taking another path between a pair of origin and destination. The goal is to design a network that satisfies the survivability with minimum total link cost. For the network design problem, we propose an efficient solution procedure based on the properties of network structure. Tabu search method is employed for the successive

improvement of the network configuration. The performance of the proposed algorithm is compared with the approach by Monma and Shallcross [9] with both sparse and dense network structures.

This paper is organized as follows. Section 2 classifies nodes and connectivities in the design of communication networks. In Section 3 the characteristics of the existing approaches are analyzed and new improvement heuristics are derived. The tabu search procedure is introduced in Section 4 to implement the heuristics developed in Section 3. In Section 5, the efficiency of the proposed algorithms is analyzed. Finally, Section 6 concludes this paper.

2. TWO CONNECTIVITY IN COMMUNICATION NETWORKS

The survivability that is considered in this paper is two node connectivity. Given a graph $G(V, E)$, suppose that each node $x \in V$ has an associated nonnegative integer connectivity type r_x. Then for any origin–destination pair of two distinct nodes $x, y \in V$, the network $N(V, F)$ where $F \subseteq E$ to be designed must contain at least $\min\{r_x, r_y\}$ node disjoint paths. The objective is to construct a minimum-cost network such that each node satisfies its node connectivity constraint. In this paper, the node set V is classified into two sets depending on the amount of traffic switched at each office. Let V_s be the set of special offices, each of which deals with relatively high amount of traffics. Also, let V_r be the set of regular offices with medium or low amount of traffic. To restore network services in the event of the complete loss of a link or the failure of switching facility, each special office requires two node connectivity. In other words, each pair of special nodes require at least two node paths between them. The two connectivity thus restores the service by routing traffic through other existing offices. On the other hand, it is enough for the regular node to have a connection to any other node in the network.

An example of communication networks which satisfy the two connectivity among special offices is shown in Fig. 1. A network $N(V, F)$ is called feasible when the connectivity constraint of each node is satisfied. Note that a feasible network that satisfies the two connectivity constraints consists of a two connected component [10] and incident spanning trees. The two-connected component is a maximal two-connected subgraph containing all special nodes. In the two-connected component there exist at least two node disjoint paths between any pair of nodes. This is illustrated by solid lines in the figure. Remaining regular nodes are connected to the two connected component as incident trees.

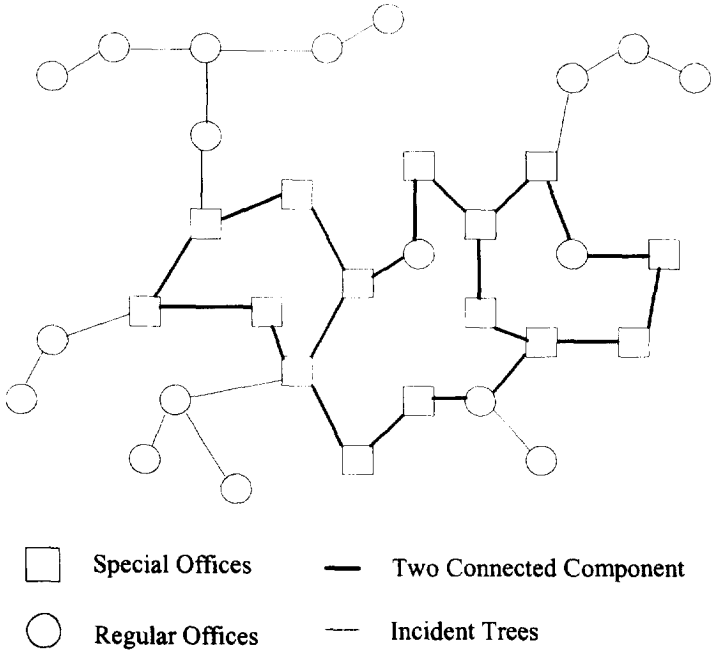

Fig. 1. Two-connectivity in a communication network.

3. IMPROVEMENT HEURISTICS

In this section, we discuss the heuristics by Monma and Shallcross [9] and present two new improvement procedures by analyzing the connectivity of the survivable network. A local improvement procedure is also provided by combining the two heuristics.

3.1. The heuristics by Monma and Shallcross

The most recently known heuristic procedure to solve the survivable network design problem is proposed by Monma and Shallcross [9]. The procedure which is called CHEUR consists of six individual improvement heuristics; two-optimal, three-optimal, pretzel, quetzel, degree and one-optimal. The CHEUR repeats these heuristics until no further cost improvement is possible.

Two-optimal and three-optimal are heuristics usually employed in the traveling salesman problem. They attempt to replace two or three links of a cycle by the other two or three links of the cycle to form a new cycle of lower cost. Each heuristic is performed by first choosing a cycle in the current solution, and then trying to interchange two or three links of the cycle. The pretzel transformation in the CHEUR replaces one link of a cycle with two crossing links to form two cycles. The quetzel transformation is the reverse of the pretzel. It transforms two cycles to one cycle by removing two crossing links and adding one link. The degree improvement heuristic reduces the degree of each special node to two or three when the link costs satisfy the triangle inequality.

All of the above five heuristics are applied only to the two connected component of the solution. The one-optimal heuristic, which considers the entire solution, attempts to remove a link (x, y) from the current feasible solution and replace it with another link of the form (x, z) not currently in the solution. Such an interchange is made only if the resultant network is feasible and of lower cost. The procedure considers each node x and all of links (x, y) incident to it as possible candidates for removal. The link (x, w) to be added to the solution is restricted among nodes which are within a predetermined distance (window size) from the node x. The one-optimal heuristic is illustrated to be the most effective among six individual heuristics. However, the procedure requires a large amount of computing time even if it can be reduced by the window size. In the following section, a more systematic heuristic is considered which maintains the feasibility depending on the links deleted.

Better solutions could also be obtained by avoiding redundancy among individual heuristics. As an example, the pretzel in some cases can be embedded in the two-optimal transformation as shown in Fig. 2a, where the square and circle nodes represent special and regular offices, respectively. Let c_{xy} be the cost of a link (x, y). If either $c_{uw} + c_{vz} < c_{uv}$ or $c_{uw} + c_{vz} < c_{wz}$, then the two-optimal heuristic in the figure corresponds to the pretzel [deleting (u, v) and adding (u, w) and (v, z)] by deleting a link (z, w). The redundancy also exists between the pretzel and one-optimal. As shown in Fig. 2b, the pretzel may include unnecessary link (u, w) which connects two regular nodes. In the following section we propose delete–add and delete–link procedures which will systematically improve the network configuration without redundancy.

3.2. Delete–add procedure

Given a feasible network $N(V, F)$, the delete–add procedure reduces the total link cost by deleting an expensive link in F and adding a cheaper link in the given link set E. This procedure

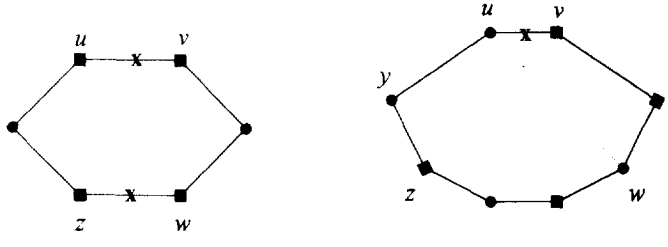

Fig. 2. Examples of the redundant transformation.

thus locally improves the solution. In some cases, however, the delete–add operation cannot be applied to the current network. Consider the following three cases with regard to the selection of candidate links to be added or deleted:

(i) No candidate links to be added exist in the link set E.
(ii) Candidate links to be added exist but their costs are not cheaper than the cost of deleted link.
(iii) No candidate links to be deleted exist due to the connectivity constraints.

In the first two cases the delete–add operation cannot be applied even if a candidate link to be deleted exists. The last case informs us that the selection of inappropriate candidate links to be deleted may lead to infeasible network structures.

Lemma 1

Given a feasible network $N(V, F)$, the delete–add operation fails to lead to a feasible network if a link (x, y) to be deleted has the following properties:

(1) $\text{degree}(x) = \text{degree}(y) = 2$.
(2) $x, y \in V_s$.

Proof

Suppose that a link (x, y) which satisfies (1) and (2) is deleted from N. Then we have $\text{degree}(x) = \text{degree}(y) = 1$. Since both x and y are special nodes, the two connectivity is not satisfied in the network even if a link (x, w) or (y, w) is added to N. QED

The above lemma implies that no link can be deleted which connects two special nodes if each node has degree two. Thus the delete–add operation is restricted to links which does not satisfy any of the two conditions of Lemma 1. For any feasible network $N(V, F)$, let T be a tree incident to the two-connected component (TCC). Clearly, the link (x, y) is contained either in a tree or in the TCC. For the construction of feasible network with the delete–add operation we consider the selection of candidate links to be added as follows.

First, suppose that (x, y) is in the tree T. When the link (x, y) is deleted, T is divided into two subtrees. Thus, a link has to be added to connect either x or y to a node which is in the other tree. Secondly, suppose that (x, y) is in the TCC. Clearly, from Lemma 1 either x or y satisfies at least one of the following conditions:

(1) $\text{degree}(x) \geqslant 3$ or $\text{degree}(y) \geqslant 3$.
(2) $x \in V_r$ or $y \in V_r$.

In the first case, since there exists an incident cycle or a tree either from the node x or y, it is easy to find a candidate link to be added. In the second case, suppose that x is a special node with $\text{degree}(x) = 2$ and y is a regular node. Since (x, y) is on a cycle, the candidate link to be added can be selected by traversing from y along the cycle until a node z appears, where z satisfies at least one of the following two conditions:

Condition 1: $z \in V_s$
Condition 2: $\text{degree}(z) \geqslant 3$.

Note that when (x, y) is deleted, to maintain the two connectivity a link (x, w) should be added where w is a node in the process of traversing up to a special node z. Therefore, as illustrated in Fig. 3 the delete–add operation is classified into the following three cases. In each case, we assume a link (x, y) is deleted and (x, w) or (y, w) is added. In case that no such links exist, the link (x, y) cannot be deleted.

Case 1. If $(x, y) \in T$, then by deleting the link (x, y), T is divided into two trees T_1 and T_2, where $x \in T_1$ and $y \in T_2$. Let T_1 be incident to TCC. Then adding a candidate link (x, w) for $w \in T_2$ or (y, w) for $w \in T_1$ or TCC results in a feasible network (see Fig. 3a).

Case 2. Suppose that $(x, y) \in$ TCC and that there exists a node z which satisfies Condition 1 in the cycle containing the link (x, y). Let P be a node set on the cycle such that

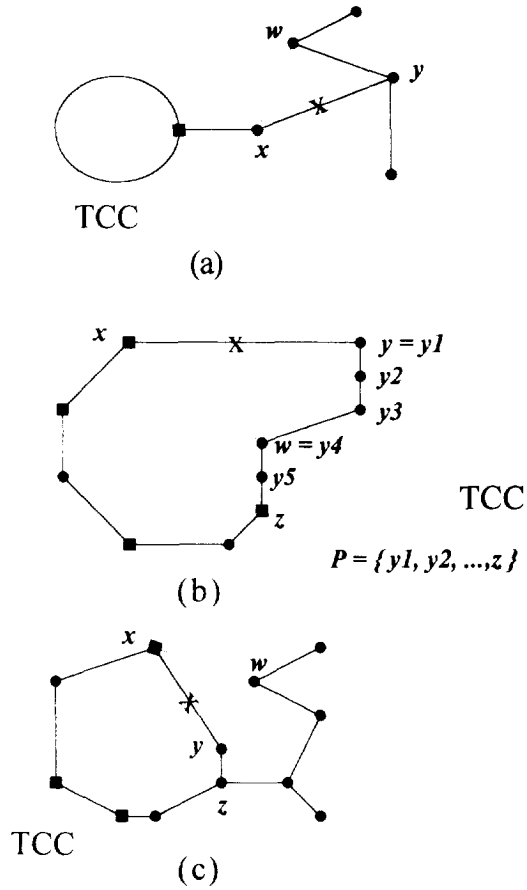

Fig. 3. Delete-add conditions.

$P = \{y_1, y_2, \ldots, z\}$. Then adding a candidate link (x, w) for $w \in P$ results in a feasible network (see Fig. 3b).

Case 3. Suppose that $(x, y) \in$ TCC and that there exists a node z which satisfies Condition 2. Then there exists an incident tree T or a cycle C from z as shown in Fig. 3c. Adding a candidate link (x, w) for $w \in T$ or C results in a feasible network.

Based on the above three cases, we present a algorithm for the delete–add procedure. The algorithm considers the priority between candidate links to be deleted by employing a heap data structure [7] for managing the current links in the network $N(V, F)$.

Delete–Add Procedure

Step 0. From a feasible network $N = (V, F)$, construct a heap H with all links in F.

Step 1. Let $l = (x, y)$ be a link with the most expensive cost in H. Set $n = m = |H|$. Update H by removing the link l. $n = n - 1$. Continue Steps 2–5 until $n \geq m/2$.

Step 2. If degree(x) = degree$(y) = 2$ and x and $y \in V_s$, then go to Step 5. Else if $l \in T$, go to Step 3. Else if $l \in$ TCC, go to Step 4.

Step 3. Select a candidate link to be added with cost improvement as in Case 1. Delete l and add the candidate link if such a link exists. Go to Step 5.

Step 4. Select a candidate link to be added with cost improvement by traversing as in the case 2 and 3. Delete l and add the candidate link if such a link exists. Go to Step 5.

Step 5. From Step 3 or 4, if there exist a candidate link with cost improvement, then insert the link into H and $n = n + 1$. Go to Step 1.

Note that we apply the delete–add operation until $n \geq m/2$. Starting from the most expensive one, one-half of the links in the current feasible network are examined by nonincreasing order of

the link cost. This is based on the assumption that the cost improvement by the rest half is not efficient enough to compensate the required computing time. The relevance of $n \geq m/2$ will be shown by the computational result in Section 5.

Compared to the one-optimal [9] which considers every link in the solution as a candidate to interchange, the delete–add procedure first prohibits the candidate link (x, y), where x and y are special nodes with degree two, from being deleted. The procedure than classifies the candidate links into three cases. In each case, the procedure searches a candidate link to be added which satisfies the feasibility condition. Thus the delete–add procedure is more effective than the one-optimal which searches a link to be added among all links within a predetermined window size from the link to be deleted.

3.3. Delete–link procedure

Another way of improving the link cost is by deleting links that are unnecessary for the feasibility. That is, links can be deleted as far as the deletion does not affect the connectivity. Suppose that an initial feasible network is constructed by two-trees [9]. Since the two-trees is constructed by adding a tree for some special nodes to the initial minimum spanning tree for all nodes in the network, there may exist links the deletion of which does not affect the feasibility.

Lemma 2

Given a feasible network $N(V, F)$, each node in V satisfies the following minimum degree conditions:

(1) For any node $x \in V_2$, degree$(x) \geq 2$.
(2) For any node $x \in V_1$, degree$(x) \geq 1$.

Proof

To satisfy the two connectivity each special node x must be on a cycle. Thus degree$(x) \geq 2$ for $x \in V_2$. Next, a regular node x needs to be connected to the network. QED

Now, to decide a candidate link to be deleted we define a redundant link of a given network. For any two nodes x, y in V, a link (x, y) in F is said redundant if and only if both x and y have degrees which are greater than the minimum of Lemma 2. That is, a link (x, y) is redundant if degree$(x) \geq 3$ for $x \in V_2$ and degree$(x) \geq 2$ for $x \in V_1$. A redundant link (x, y) in F can be deleted if one of the following conditions is satisfied:

Condition 1. For x and $y \in$ TCC, there exists a cycle containing x and y in $N[V, F\backslash(x, y)]$.
Condition 2. For x and y such that $x \notin$TCC or $y \notin$TCC, there exists a path from x to y in $N[V, F\backslash(x, y)]$.

Figure 4 illustrates the above two conditions on which delete–link procedure is based. The procedure also considers the priority between candidate links to be deleted by employing a heap data structure to manage the links in $N(V, F)$.

To identify a cycle or a path in the above conditions, a depth-first tree is constructed which spans all nodes in F starting from the node x. If there exists a cycle containing the two nodes x and y, then the starting node x will be revisited from the node y on the tree. The existence of a path between two nodes x and y can also be identified by the depth-first tree. Two nodes are connected if node y is reachable on the tree from node x. The identification of a cycle or a path can be performed in a running time of $O(m)$, where m is the number of links in F.

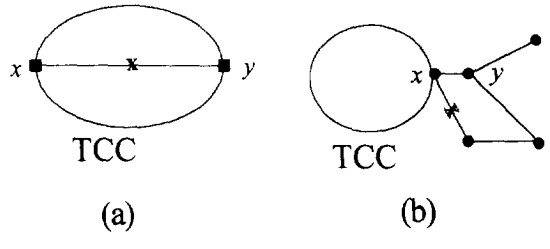

Fig. 4. Delete–link conditions.

Delete–Link Procedure

Step 0. Construct a heap H with all edges in F.
Step 1. Let l be a link with the most expensive cost in H. Update H by removing l. Continue Steps 2–3, until H will be empty.
Step 2. If l is a redundant link, then go to Step 3, else go to Step 1.
Step 3. If l satisfies one of the two conditions, then remove l from F. Go to Step 1.

As shown in Lemma 2, the delete–link procedure improves the network cost by deleting redundant links. Since the redundancy may occur either in the two connected component or in the remaining part of the network, the procedure scans all links in the two parts. However, the degree heuristic [9] considers links only in the two connected component. Thus it may include unnecessary links in the solution.

3.4. Local improvement procedure

In previous sections, we have proposed two improvement heuristics. The delete–add procedure exchanges a link with the less expensive one, while the delete–link procedure just excludes unnecessary links. To generate a more improved solution, we present a local improvement procedure which combines the two heuristics with the two-optimal procedure. The two-optimal heuristic [9] is applied to links in the two-connected component. It attempts to replace two links of a cycle by two other links of the cycle to form a new cycle of lower cost. We first choose a cycle in the current solution, and then try to interchange two links of the cycle. This step is repeated until no further improvements are possible.

Local Improvement Procedure

Step 0. Get an initial feasible network.
Step 1. Perform the delete–add procedure. Let Opt1 be the link cost of the network.
Step 2. Perform the two-optimal procedure. Let Opt2 be the link cost of the network.
Step 3. If Opt2 < Opt1, then go to Step 1, else go to Step 4.
Step 4. Perform the delete–link procedure. Let Opt3 be the link cost of the network.
Step 5. If Opt3 < Opt2, then go to Step 1, else stop.

4. TABU SEARCH FOR THE SURVIVABLE NETWORK DESIGN

Tabu search [3, 4] is a search technique which allows uphill move to overcome local optima. At each step, the neighborhood of the current solution is explored and the best solution is selected as the new current solution. This procedure is called a "move". However, as opposed to other local search techniques, the procedure does not stop even when no improvement is obtained. The best solution in the neighborhood is selected, even if it is worse than the current solution. This strategy allows the search to escape from local optima and to explore a larger fraction of the search space. To prevent cycling in the search process, recently selected solutions are forbidden and inserted into a "tabulist". For each move leading to a new solution, the inverse move is labeled "tabu" and stored in the tabulist.

4.1. Design of move

In most of tabu search applications for network problems, a move is usually defined as to delete a link in a current solution and add the best candidate link among those which satisfy the problem constraints. Such a move can be considered as a *micro* move which corresponds to the individual improvement heuristic presented in Section 3. In this paper, we employ the local improvement procedure of Section 3.4 as the *macro* downhill move. The uphill move is allowed by constructing a new feasible solution from the current solution. The construction of a new feasible solution is performed by deleting a link in a cycle and adding two crossing links as illustrated in Fig. 5.

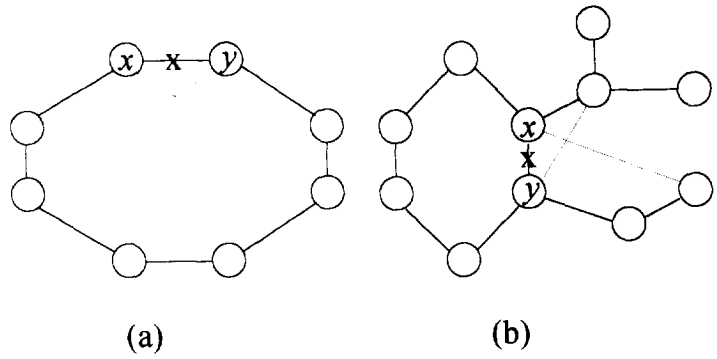

Fig. 5. Uphill move procedure.

Uphill Move Procedure

Step 0. Let $N(V, F)$ be a current solution by a downhill move.
Step 1. Find the most expensive link in a cycle C of $N(V, F)$.
Step 2. Delete the most expensive link (x, y) in C.
Step 3. From each node x and y, add a link by connecting a node in the cycle C such that the two links are crossed (see Fig. 5a). If no such links are found in C, then add links which connect nodes in the given graph $G(V, E)$ (see Fig. 5b). Stop.

Note that the new network constructed by the uphill move always satisfies the connectivity constraints of the given problem. It thus generates a feasible network even if the total link cost is increased.

4.2. Tabulist and tabu parameters

The tabulist consists of the links deleted in Step 2 of the uphill move procedure. These links are restricted from being added in the following downhill move such as the delete–add and two-optimal procedures.

The application of the tabu search employed in this study requires two parameters to be considered: tabusize and N_{max}. Tabusize represents the maximum number of links in the tabulist. Each time a link is included in the list, the tabulist is updated by freeing the oldest link. The N_{max} relates to the stopping rule. The search stops when there is no cost improvement for N_{max} consecutive iterations. Computational experiments are performed to determine the tabusize as illustrated in Fig. 6. Each value in the figure represents the ratio of the network cost. From the

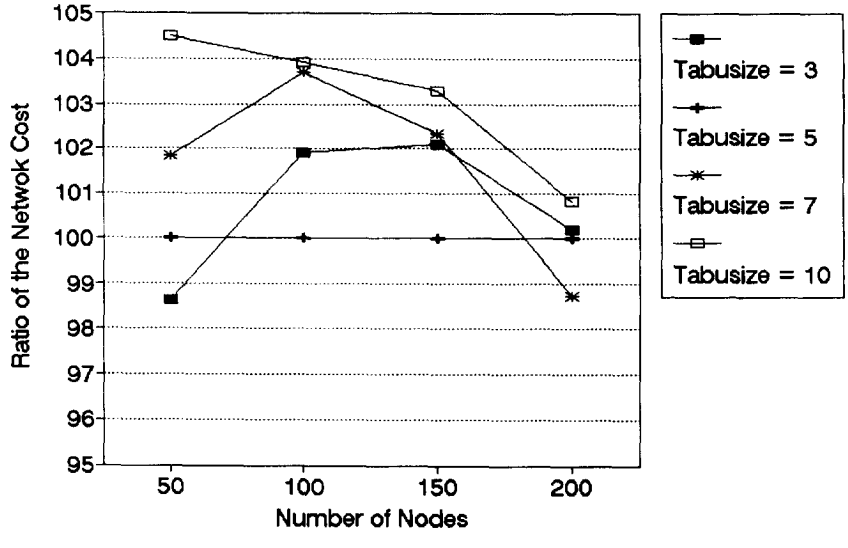

Fig. 6. Determination of tabusize.

figure it seems to be reasonable to keep a different tabusize in each problem. Based on the preliminary test, the tabusize is determined as three for problems with 50 nodes, five for 100 and 150 nodes and seven for 200 nodes. For the stopping rule N_{max}, almost same result is obtained for $N_{max} = 3$ and $N_{max} = 5$. Thus we suggest to stop the search when there is no cost improvement for three consecutive tabu iterations.

4.3. *Tabu search procedure*

Based on the previous discussion, we present a tabu search procedure for the survivable network design problem. Each move consists of a downhill move and an uphill move.

Initialization Phase

Step 0. Construct an initial two connected network. Initialize the tabulist. $N_{max} = 3$ and num = 0;

Phase I (Downhill Move)

Step 1. Perform the local improvement procedure of 3.4 by checking tabu status.
Step 2. **IF** a new best solution is found by the procedure **THEN** num = 0.
ELSE num = num + 1.
Step 3. **IF** num $\leqslant N_{max}$ **THEN** go to Phase II. **ELSE** stop.

Phase II (Uphill Move)

Step 4. Construct a network by the uphill move procedure of 4.1.
Enroll the deleted link into tabulist. Go to Phase I.

5. COMPUTATIONAL RESULTS AND DISCUSSION

To test the performance of algorithms proposed in this paper a set of problems with 50, 100, 150 and 200 nodes are generated for both dense and sparse networks. For each problem, five instances are generated by placing nodes in the 300×300 plane. Euclidean distances are used as the link costs. Each node is randomly chosen to be either a regular or a special node with equal probability. A complete graph is employed for the underlying graph of the dense problem, while that of the sparse problem is generated such that the degree of each node becomes 5, 6 or 7 starting from the two-trees [9] feasible network.

All the computational results reported here are performed on IBM-PC/AT 486. The programs is implemented in C language. The CPU times reported exclude the time required to initialize the data structures. The result of the computational tests on the 40 randomly generated problems are summarized in Tables 1–5. Each value shown in the table represents the average solution. The values in the parenthesis represent CPU seconds.

We first demonstrate the relevance of $n \geqslant m/2$ in the delete–add procedure which restricts the candidate links to be deleted to the most expensive one half in the current feasible network. Figure 7 shows the cost improvement by delete–add procedure when $n \geqslant 3m/4$, $n \geqslant m/2$ and $n \geqslant m/4$. It is clear from the figure that the cost improvement is remarkable when $n \geqslant m/2$. Increasing the number of candidate links as in $n \geqslant m/4$ is not so promising when we consider the computing times required for the additional improvement.

Computational results of the one-optimal and the delete–add are shown in Tables 1 and 2. No restriction is applied to the window size in implementing the one-optimal heuristic. Thus the result of the one-optimal is the best solution that can be improved by the heuristic. From the two tables it is clear that the delete–add procedure gives better solution that the one-optimal heuristic both in dense and sparse problems.

Now, for the implementation of individual heuristics in CHEUR we employ three different orders. First, the CHEUR is implemented in the order of one-optimal, quetzel, three-optimal, two-optimal, degree and pretzel which is the general descending order of improvement range. Secondly, individual heuristics are implemented in ascending order. Finally, the following order is employed; one-optimal, three-optimal, two-optimal, pretzel, quetzel and degree, where the first

Table 1. Comparison of one-optimal and delete-add in dense problems

No. of nodes	Initial two-trees	One-optimal	Delete-add
50	2134 (0.16)	1913 (1.36)	1832 (0.48)
100	3038 (0.72)	2592 (9.60)	2558 (3.40)
150	3632 (2.60)	3116 (28.14)	3082 (8.60)
200	4042 (6.60)	3491 (82.90)	3442 (21.32)

Table 2. Comparison of one-optimal and delete-add in sparse problems

No. of nodes	Initial two-trees	One-optimal	Delete-add
50	2134 (0.16)	1957 (0.66)	1947 (0.20)
100	3038 (0.72)	2654 (4.80)	2629 (1.60)
150	3632 (2.60)	3220 (13.00)	3139 (4.60)
200	4042 (6.60)	3584 (31.20)	3540 (10.64)

Table 3. Optimal solutions of problems with 40 nodes

Problem No.	CHEUR	Tabu search	Optimum	(CHEUR-OPT) × 100/OPT (%)	(TS-OPT) × 100/OPT (%)
Instance 1	2186	2186	2175	0.51	0.51
Instance 2	2149	2136	2117	1.51	0.90
Instance 3	2178	2152	2152	1.21	0.00
Instance 4	1981	1976	1961	1.02	0.76
Instance 5	2148	2148	2148	0.00	0.00
Average	2128	2120	2111	0.85	0.43

Optimal solutions are obtained by the CPLEX.

Table 4. Computational results in dense problems

No. of nodes	CHEUR	Tabu search	(CHEUR-TS) × 100/TS (%)
50	1671	1581	5.69
	(4.41)	(7.24)	
100	2321	2223	4.38
	(34.83)	(41.48)	
150	2804	2703	3.71
	(112.31)	(135.42)	
200	(3112)	3011	3.33
	(282.64)	(315.40)	

Table 5. Computational results in sparse problems

No. of nodes	CHEUR	Tabu search	(CHEUR-TS) × 100/TS (%)
50	1744	1700	2.61
	(2.06)	(2.48)	
100	2386	2327	2.54
	(17.40)	(22.40)	
150	2876	2808	243
	(65.80)	(77.40)	
200	3241	3180	1.92
	(145.20)	(196.20)	

four heuristics improve the link cost by maintaining the number of links and the other methods by reducing it. We choose the best solution of three methods as the performance of CHEUR.

To justify the implementation of CHEUR, optimal solutions are obtained for five instances. Each instance has 40 nodes and 90 links. The cutting plane algorithm [6] is employed to solve the integer linear problem. CPLEX is used to solve the LP problem whenever valid inequalities are added. Table 3 shows the result of CHEUR and tabu search. Note that Groetschel *et al.* [6] solved the two connectivity problem with 39 nodes and 86 underlying potential links (in view of reduced graph). They showed that the percent relative error from the optimum is below 1.5%. The table

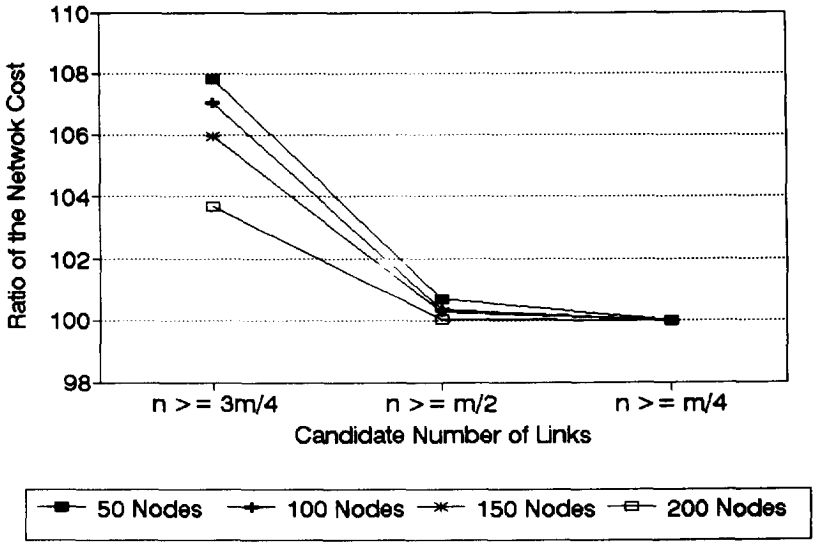

Fig. 7. Comparison of delete–add.

illustrates that CHEUR generates a very effective solution. The average gap from the optimum is 0.85%, which is very close to the result by Ref. [6].

Finally, Tables 4 and 5 show the performance of the tabu search and the CHEUR method. From the tables, the tabu search is illustrated to be superior to the method by CHEUR. The effectiveness of the tabu search is more prominent in dense problems than in sparse problems. On the average, 3–6% cost reduction effect is obtained by the tabu search technique. Figure 8 shows an example tabu search with a 100-node problem. Successive uphill and downhill move is illustrated with an escape from local optimum.

6. CONCLUSIONS

Two improvement heuristics, delete–add and delete–link procedures, are developed to solve the communication network design problem with two connectivity constraints. In the delete–add procedure, each candidate link is selected such that it constructs a tree or a cycle depending on

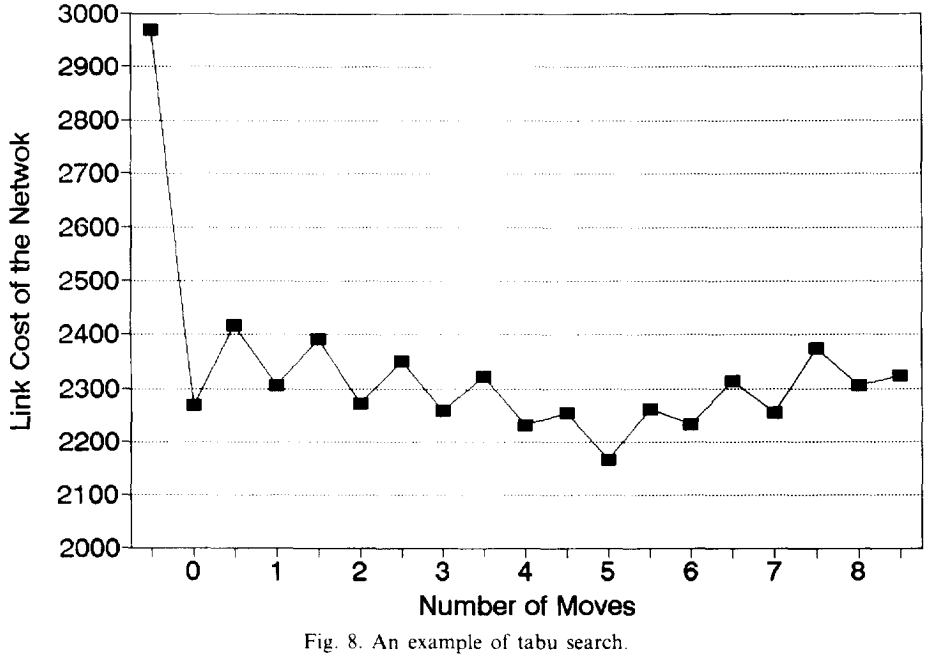

Fig. 8. An example of tabu search.

the status of the deleted link. The delete–link procedure, on the other hand, deletes a link as far as the two connectivity is satisfied even after the deletion. A local improvement procedure is presented by combining the two improvement heuristics with the two-optimal procedure.

A tabu search method is proposed to improve the solution by overcoming the local optima. Uphill moves are allowed by exchanging the most expensive link with two crossing links such that each link constructs a separate cycle in the network.

The proposed tabu search is proved to avoid the local optima effectively. The solution obtained by the search process outperforms that by the CHEUR which is the best known method in the design of survivable networks. On the average, 3–6% cost reduction effect is obtained by the tabu search method.

REFERENCES

1. D. Bienstock, E. F. Brickell and C. L. Monma. On the structure of minimum-weight k-connected spanning networks. *SIAM J. Discr. Math.* **3**, 320–329 (1990).
2. R. H. Cardwell, C. L. Monma and T. H. Wu. Computer-aided design procedures for survivable fiber optic networks. *IEEE J. Selected Areas Commun.* **7**, 1188–1197 (1989).
3. F. Glover. Tabu search—Part 1. *ORSA J. Comput.* **1**, 190–206 (1989).
4. F. Glover. Tabu search—Part 2. *ORSA J. Comput.* **2**, 4–32 (1990).
5. M. Groetschel and C. L. Monma. Integer polyhedra associated with certain network design problems with connectivity constraints. *SIAM J. Discr. Math.* **3**, 502–523 (1990).
6. M. Groetschel, C. L. Monma and M. Stoer. Computational results with a cutting plane algorithm for designing communication networks with low connectivity constaints. *Oper. Res.* **40**, 309–330 (1992).
7. E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithms*. Computer Science Press (1978).
8. C. L. Monma, B. S. Munson and W. R. Pulleyblank. Minimum-weight two-connected spanning networks. *Math. Program.* **46**, 153–171 (1990).
9. C. L. Monma and D. F. Shallcross. Methods for designing communications networks with certain two-connected survivability constraints. *Oper. Res.* **37**, 531–541 (1989).
10. R. Tarjan. Depth-first search and linear graph algorithm. *SIAM J. Comput.* **1**, 146–160 (1972).