Annals of Operations Research 33(1991)351–362

LEAST-COST NETWORK TOPOLOGY DESIGN FOR A NEW SERVICE: An application of tabu search*

Fred GLOVER
Center for Applied Artificial Intelligence, University of Colorado, Boulder, CO 80309-0419, USA

Micheal LEE
Public Service Company of Colorado, Denver, CO, USA

and

Jennifer RYAN
Department of Mathematics, University of Colorado at Denver, Denver, CO 80217-3364, USA

Abstract

We describe an implementation of the tabu search metaheuristic that effectively finds a low-cost topology for a communications network to provide a centralized new service. Our results are compared to those of a greedy algorithm which applies corresponding decision rules, but without the guidance of the tabu search framework. These problems are difficult computationally, representing integer programs that can involve as many as 10,000 integer variables and 2000 constraints in practical applications. The tabu search results approach succeeded in obtaining significant improvements over the greedy approach, yielding optimal solutions to problems small enough to allow independent verification of optimality status and, more generally, yielding both absolute and percentage cost improvements that did not deteriorate with increasing problem size.

1. Introduction

We describe an empirical study using tabu search to find a low-cost topology for a communications network to provide a centralized new service. Such a new service may be a phone-in service to access data, or more broadly may involve any application where the object is to connect the users to the service, rather than to each other.

The following section presents the problem formulation. In section 3, we describe a method based on the tabu search metaheuristic to solve this problem. Section 4 discusses the results, comparing them to those of a greedy algorithm which applies corresponding decision rules, but without the guidance of the tabu search framework.

*This research was partially supported by the Air Force Office of Scientific Research and the Office of Naval Research Contract No. F49629-90-C-0033.

2. The new service topology design problem

The problem under study is that of finding a low-cost design for a communications network that introduces a new service. The problem derives from a practical application in the telecommunications industry whose importance is underscored by the US West commissioned study reported by Ryan [8] (see also Parrish et al. [7]). The new service at the core of this problem is provided by computers located in certain switching offices called *platforms*. There may be more than one way to set up a platform. Generally, a higher capacity platform has a higher installation cost. Demand for the service can originate at the platforms or at any other switching office. An entire new set of circuits must be built to carry the new traffic, due to regulatory restrictions on the use of the circuits of the existing network.

Many practical applications involve just two types of circuits, consisting of voice-grade circuits and digital circuits. Our algorithm solves a general version of the problem, where any number of circuit types may be available. Each type of circuit has associated with it an installation cost and an operational cost. The installation cost is a one-time fixed cost, and the operational cost is an ongoing variable cost that depends on the volume of traffic using the circuit.

The network design must specify at which switching offices the platforms should be located, and what type of platform should be installed at each site. The design must also determine where the various types of circuits should be installed.

To formulate the problem mathematically, let P denote the set of platform types and L the set of circuit types. Each platform type $p_i \in P$ has two associated parameters: a platform installation cost p_c and a platform service capacity p_{cap}. With each of the available circuit types $l \in L$ we associate the parameters l_{ic}, l_{oc} and l_{cap}, which identify the circuit installation cost per unit distance, the circuit operational cost per unit flow and the circuit capacity.

A potential solution to the problem may be represented by a pair (V, E), where V is the set of nodes to be serviced and E is the set of ordered pairs, or edges, which can be a site for one or more circuits. Each $v \in V$ has the following parameters associated with it:

v_d demand generated by v,

v_{in} demand flowing into v (for service or transshipment),

v_{out} demand flowing out of v,

v_c demand servicing capacity of v,

v_{P_i} number of platforms of type i at v.

All of the above parameters are nonnegative integer variables to be given values by the problem solution except for v_d, which is part of the original problem data (assumed positive and integer). Note that unless v has been designated a platform site, v_{P_i} is 0 for each i and v_c is 0. Further, the difference $v_{in} + v_d - v_{out}$ must be nonnegative

and must not exceed v_c. The parameter v_d is the only constant in the above list. The current servicing capacity of a node v_c is a function of the values of the v_{p_i}'s.

Each $e = (i, j) \in E$ is associated with the following parameters:

e_{d_i} demand flowing from vertex i to vertex j,

e_{d_j} demand flowing from vertex j to vertex i,

e_c total demand that can be carried on e,

e_l set of circuit types which make up e.

The value of e_c is a function of e_l and the capacities of the circuits in e_l. Also, at most one of e_{d_i} and e_{d_j} is positive.

Finally, for each pair of nodes i and j in V, we denote by d_{ij} the distance from i to j. The distance d_{ij} is a constant, independent of the current solution; it is the length of edge $e = (i, j)$ if such an edge exists, and is ∞ otherwise. The cost of installing a link between nodes i and j depends on d_{ij}. In real terms, it affects the cost of digging the ditch to install the cable. Again, all values are assumed to be nonnegative integers.

The total cost W to be minimized is composed of the total edge costs W_E and the total node costs W_V, which are calculated as follows:

$$W_V = \sum_{v \in V} \sum_{p_k \in P} v_{p_k} p_{k_c};$$

$$W_E = \sum_{e=(i,j) \in E} \sum_{l_k \in e_l} (l_{k_{ic}} * d_{ij} + l_{k_{oc}} * flow_{e_k}),$$

where $flow_{e_k}$ is the flow on the circuit of type k on edge e. For edge $e = (i, j)$, the quantity $flow_{e_k}$ is easily determined by considering $e_{d_i} + e_{d_j}$ (the total flow on edge e) and e_l. The various circuits in e_l are used greedily, that is, the links with the smallest operational cost will be saturated before a more expensive link is used. Our algorithm ensures that there will be no unused links in e_l.

Thus, the new service topology design problem is to minimize W subject to the following constraints:

- the edges with e_l nonempty must form a connected network;
- for each $v \in V$, $0 \le v_d + v_{in} - v_{out} \le v_c = \sum_{i \in P} v_{p_i} p_{i_{cap}}$;
- for each $e = (i, j) \in E$, $e_{d_i} + e_{d_j} \le e_c = \sum_{l \in e_l} l_{cap}$.

We denote a particular solution, which includes the assignment of platforms, circuits, and a feasible flow, by x. The cost of this solution will be denoted $W(x) = W_V + W_E$.

3. The heuristic approach

Our approach for finding the least-cost topology design is divided into two component phases which are applied in alternating succession. The first phase consists of a *platform location routine*, which assigns the platform locations and tentatively assigns circuits and platform capacities by routing the demand at each node to the nearest platform. The second phase consists of a *circuit assignment routine*, which takes the output from the first phase, assigns platform types to the platform locations, and makes local improvements by testing for each node v the economy of routing any of the flow to the neighbors of v through v. Each phase has its own associated tabu search control parameters.

Hertz and de Werra [4] have noted that tabu search functions more effectively when the topography of the solution space is not too "flat", i.e. where the terrain does not induce the search to visit long successions of solutions whose objective function values are very similar. Several methods have been suggested to improve performance when such regions are encountered. Glover [2] has presented the concept of *strategic oscillation*, which drives the search to progress for selected distances beyond "boundaries", such as specified by objective function levels or measures of feasibility, and then to drive "in reverse" to meet and cross the boundary from the opposite side. Such an oscillatory approach either directly or indirectly produces greater fluctuations in the objective function and generally avoids the symptomatic "flat trajectory" behavior. Hertz and de Werra [4] suggest altering the formulation of the problem in such a way that all feasible solutions meet the desired objective value, and create a new objective function that measures infeasibility.

In this study, fluctuations in the objective function are strategically induced by nesting the two tabu search phases in a particular way. Specifically, a complete solution pass of the second phase is executed for each iteration of the first phase. That is, each time the first routine makes a "move", which in general is a nonimproving move, the second routine attempts to reoptimize.

The main structure of the outer loop, which creates the platform location assignments and other tentative associated assignments of the first phase, is as follows.

Let x = initial solution;
Let $x^* = x$;
Let $T = \{\ \}$;
Repeat
  Let $S(x)$ = **Neighborhood**$_1(x)$;
  Let x' = **Optimize**$_1(S(x), T)$;
  If $W(x') < W(x)$ then
    Let $x^* = x'$;
  Let $x = x'$:
Until (**Stable**$_1(x)$)

The set T is the tabu list, which is initially empty. The function **Neighborhood$_1$** returns a set of transformations which will vary the distribution of the platform capacity in the network. The function **Optimize$_1$** takes the set of possible solutions given by the neighborhood function and calculates the next trial solution, taking the tabu status of the possible moves into consideration. **Optimize$_1$** also performs the update to the tabu list and calls the routine to solve the circuit assignment problem. We now describe each of these functions in detail.

There are two types of transformations returned by **Neighborhood$_1$**. One distributes the platform capacity to nodes which have no platforms, the other undertakes to centralize the platform capacity at a certain node. The function **Neighborhood$_1$** is sketched below.

Let $S(x) = \{\ \}$;
For all $i \in V$ with $i_c \neq 0$
  For all $j \in V$ with $j \neq i$
    if $e = (i, j) \in E$ and $e_c > 0$ then
      if $j_c = 0$ then
        $S(x) = S(x) \cup \{(1, j, i, \alpha_{ij})\}$;
      else
        $S(x) = S(x) \cup \{(2, j, i, \beta_{ij})\}$;
      endif;
    endif;
  endfor;
endfor;

where

1 means distribute capacity from i to j;

2 means centralize capacity from j to i;

α_{ij} is an approximate cost of distributing capacity from i to j;

β_{ij} is an approximate cost of centralizing capacity from j to i.

The cost of distributing capacity from i to j is approximated by

α_{ij} = (cost of entire network per demand serviced)
  – (average cost of demand serviced at i).

The cost of centralizing capacity from j to i is approximated by

β_{ij} = (average cost of demand service at i)
  – (average cost of demand serviced at j).

The **Optimize$_1$** function calculates the next trial move by applying the best m transformations returned by **Neighborhood$_1$**, where m is a parameter to be fixed

before running the program. (We have thus created a "candidate list" of moves, see Glover [3].) Before calling the routine to solve the circuit assignment problem, **Optimize$_1$** greedily routes the demand of each node to the nearest platform. A sketch of the function **Optimize$_1$** is given below.

Partially order the set $S(x)$ in order of increasing cost.
Apply the first m transformations (σ, j, i, δ) as follows:
  if $(\sigma, j) \notin T$ then
    if $\sigma = 1$ then
      add a platform to vertex j;
      Let $T = T \cup \{(2, j)\}$;
    else ($\sigma = 2$)
      delete all platforms at vertex j;
      let $T = T \cup \{(1, j)\}$;
    endif;
  endif;
Clear all routings;
RouteDemand;
Optimize$_2$(Neighborhood$_2$(x));

The RouteDemand procedure greedily routes all demand to the nearest platform, making tentative circuit assignments to carry that flow. Platforms which provide excess capacity after an iteration of **Optimize$_1$** are removed in the routine **Optimize$_2$**.

The implementation of the tabu list T above, which has the function of excluding certain pairs (σ, j) from consideration during the search, is critical. One of the goals of a tabu list is to prevent cyclic behavior. The simplest forms of such lists are designed chiefly to prevent the reversal of moves made on recent iterations.

However, a high degree of specificity in characterizing the moves that are prevented from being reversed can be counterproductive relative to maintaining an appropriate balance of diversity in the solutions generated (see, for example, Glover [2], Malek et al. [6], Skorin-Kapov [9]). Consequently, it is important to isolate appropriate *attributes* of moves as a basis of defining tabu status (membership on T), which have the effect of preventing certain classes of moves rather than certain specific moves from being performed, for the duration that the associated tabu status remains in effect. Our choice of attributes consisting of the pairs (σ, j), which proved much more effective than attributes designed to prevent more specific move reversals, can be explained by the following reasoning. Suppose that i, j and k are elements of V, and that one of the most recent moves consisted of centralizing capacity by a shift of capacity at j to node i. Then the move to distribute capacity from k to j (rather than the narrower "reversal" from i to j) should be prohibited because capacity had just been deleted from j and would probably end up being centralized at i again. After the move which centralizes capacity from i to j is made, $(1, j)$ is placed on the tabu

list, which has the desired effect of prohibiting a move to distribute capacity from k to j for any k. Similarly, after a move which distributes capacity from i to j is made, $(2, j)$ is placed on the tabu list to prohibit the centralization of capacity from j to any other node k. Empirically, this intuitively-based determination of tabu attributes was found to work very well.

The circuit assignment routines **Neighborhood$_2$** and **Optimize$_2$** make local improvements in the circuit sizing in an attempt to reduce costs. The circuit assignment neighborhood function **Neighborhood$_2$** creates a set of two types of moves. The first type of move will add the different platform types to nodes with platforms. The second type of move seeks to optimize the circuit types on the edges which carry demand. This is done heuristically by checking for each node v and each neighborhood w of v whether costs can be decreased by routing the flow to w through v. If such a change is made, the tentative circuit assignments are altered to reflect that change.

The circuit assignment optimization function **Optimize$_2$** then continuously applies the moves selected by **Neighborhood$_2$** until a fixed number of moves have been made. When new circuits are added to an edge, or new platforms are added to a vertex, older circuits and platforms which are in excess of the necessary capacity are automatically deleted. An additional simple tabu list is used to prevent cycling in this process by straightforwardly designating the identities of platforms and circuits deleted to be the attributes for defining tabu status for addition. Following suggestions of Glover [2], we experimented with tabu list lengths in the range 5 to 9. Tabu lists of length 7 or 8 were found to work well for both the platform location routine and the circuit assignment routine.

4. Results

The algorithm described in section 3 was implemented using Turbo C 2.0 on an IBM AT compatible microcomputer. Random test cases were generated for problems containing 5, 6, 10, 25, 40, and 50 nodes. Because of the relatively recent identification of the practical import of this problem (Ryan [8]), data from specific applications are unavailable, but the dimensions examined subsume the range to be expected in such applications. It is to be noted that a 25-node problem corresponds to an integer program with 2575 variables and 650 constraints, while a 50-node problem corresponds to an integer program with 10,150 variables and 2550 constraints (using three circuit types and three platform types, which were the parameters of our test problems). The method always found known optimal solutions to the five- and six-node test cases (corresponding to integer programs with 162 variables and 42 constraints). Due to the combinatorial complexity of the problem, optimal solutions to the 10-, 25-, 40- and 50-node test cases are not known.

To provide a basis for relative evaluation, results for these cases have been compared to a greedy algorithm which assigns a fixed number of platforms to those nodes having greatest demand, and routes the flow by the same process used in the outer loop of our algorithm. This "greedy" method of platform location was used by

Anderson and Rasmussen [1]. The algorithm of Parrish et al. [7] also requires that a single platform site is chosen in advance.

Average running times for our method and the greedy method are given in table 1. The first column indicates the average time to find the greedy solution. The second and third columns give the average time to find the best solution in the tabu

Table 1

Average running times for test problems

Test case	Avg. time for greedy	Avg. time to best for tabu	Avg. tot. time for tabu
5 node	–	–	30 seconds
6 node	–	–	30 seconds
10 node	25 seconds	42 seconds	2 minutes 19 seconds
25 node	1 minute 31 seconds	11 minutes 55 seconds	19 minutes 51 seconds
40 node	2 minutes 8 seconds	18 minutes 45 seconds	46 minutes 28 seconds
50 node	3 minutes 15 seconds	25 minutes 40 seconds	2 hours 2 minutes

search, and the average time the tabu search ran, respectively. The greedy method was not tested on the 5- and 6-node problems. Solution value comparisons with the baseline greedy method are given in table 2. Sample performance graphs are given

Table 2

Comparison of test results with greedy solution

Test case		Greedy solution	Tabu search solution	Percent improvement
10 node	1	124094	101182	18.5
	2	110702	98941	10.6
	3	115074	99576	13.5
25 node	1	262874	211470	19.6
	2	264986	213867	19.3
	3	263143	217149	17.5
40 node	1	409076	333441	18.5
	2	395973	333274	15.8
	3	398089	333080	16.3
50 node	1	529842	417079	21.3
	2	562618	409024	27.3
	3	526636	427000	18.9
	4	536319	414894	22.6
	5	510631	415931	18.6

in figs. 1 and 2. The spikes correspond to an iteration of the platform assignment routine. Figures 3 and 4 illustrate pictorially the input and the output of the algorithm. As shown in table 2, the tabu search approach performed significantly better than the

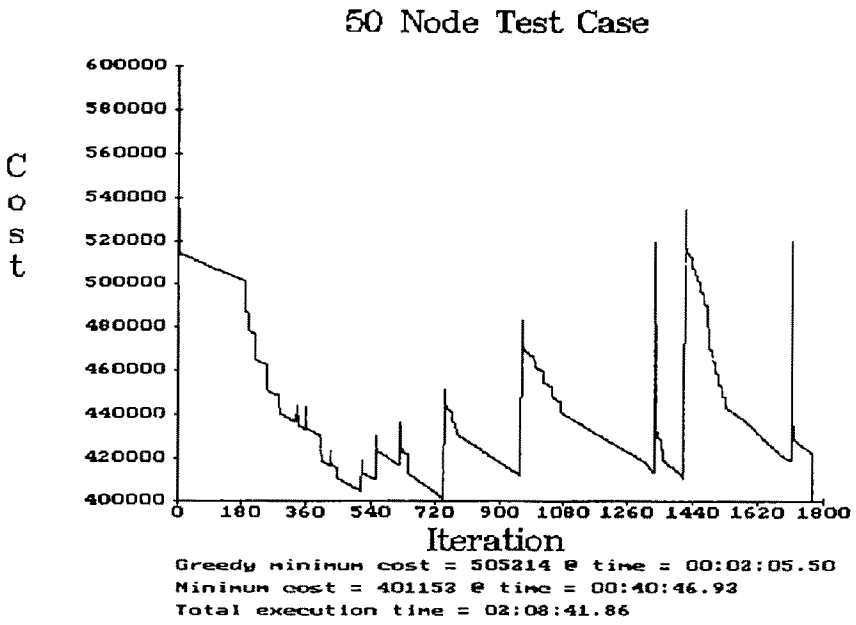


Fig. 1. Sample performance graph.

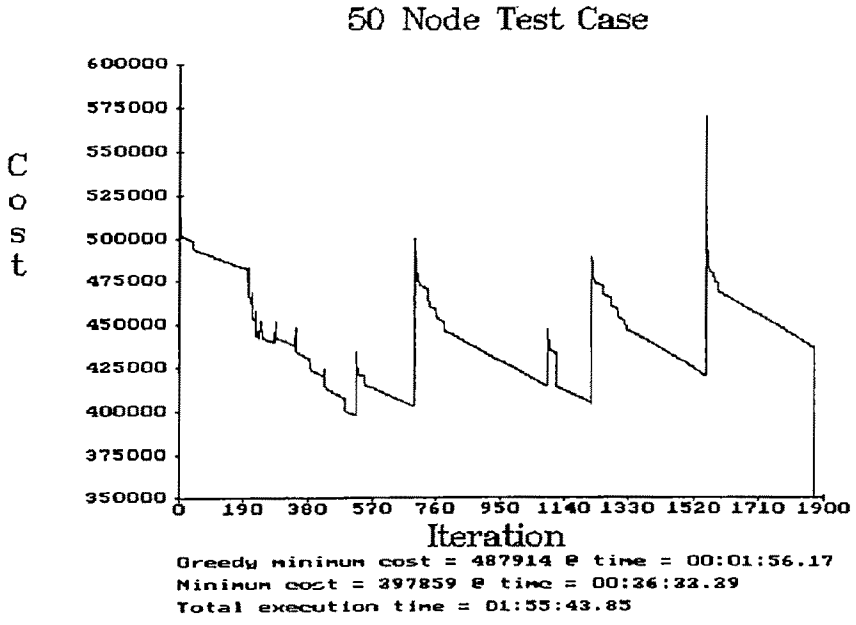
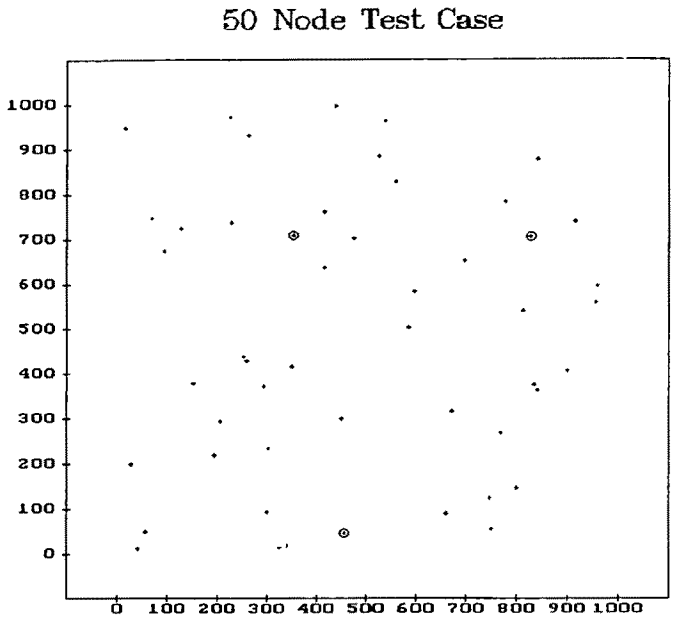
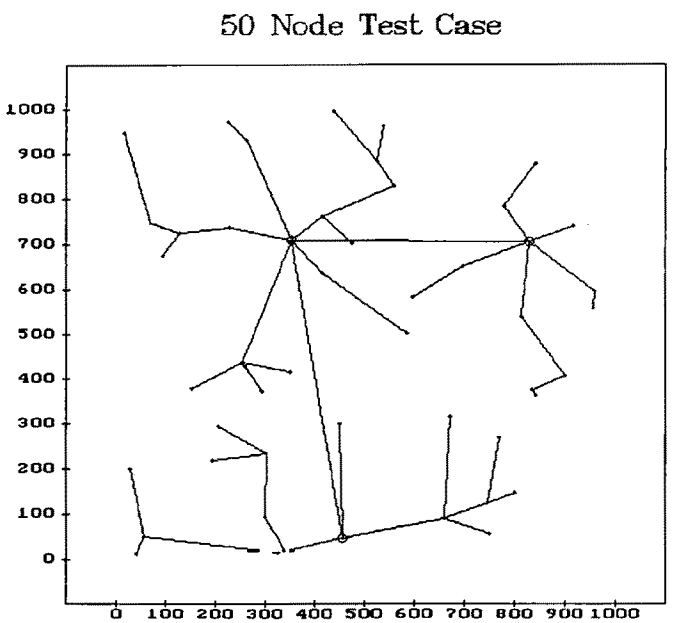


Fig. 2. Sample performance graph.

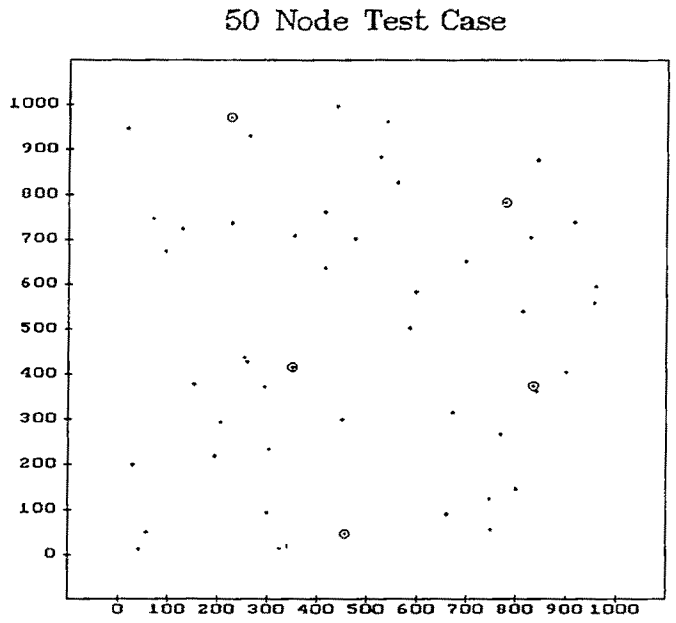


(a)

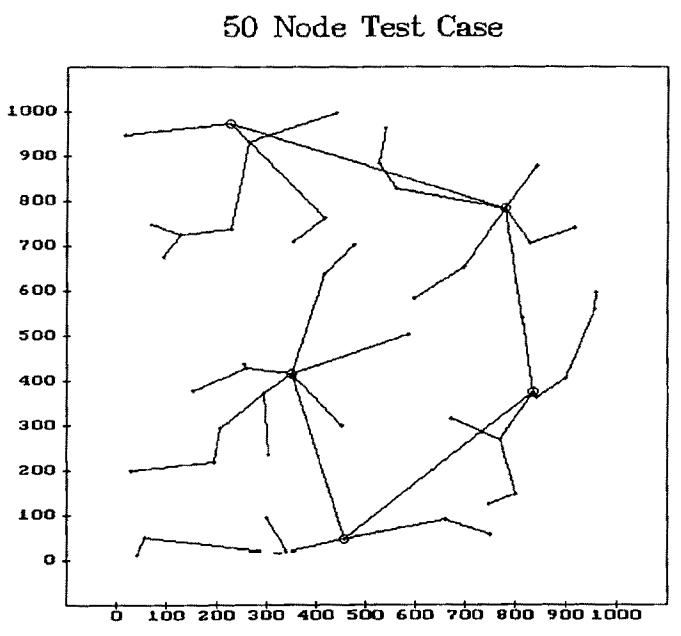


(b)

Fig. 3. Input (a) and output (b) of the algorithm.



(a)



(b)

Fig. 4. Input (a) and output (b) of the algorithm.

greedy procedure. The differences identified in the table, which identify cost improvements that average greater than 15% (and above 20% for the 50-node problem), translate into significant financial savings in practical settings [7].

Some general patterns were observed in the solutions found. When the costs of platforms were very low in relation to the circuit operational costs, platforms were distributed to every node and a minimum amount of routing was performed to maintain connectivity. When the costs of the platforms were higher, a correspondingly greater degree of centralization occurred.

Several possible avenues for improvement are apparent. The evaluation of the network at each iteration is only approximate; there is no guarantee that the flows based on each chosen assignment of circuits and platforms are determined optimally. The application of a minimum cost flow routine would give an exact evaluation of the network. Although this would probably be too expensive to perform at every iteration, it may be worth doing every "few" iterations", and in particular as a means of refining the candidates for best solutions generated by the method in its present form.

Minor improvements were obtained in some cases when the routines were made more aggressive by making more than one move per iteration. An attempt to identify the "optimal" number of moves per iteration in early runs could be the basis for a strategy to allow faster solutions of later runs.

References

[1] C. Anderson and C. Rasmussen, A genetic algorithm for the new service network topology problem, in: *Heuristics for Combinatorial Optimization*, Section 7 (1989), pp. 1–23.

[2] F. Glover, Tabu search, Part 1, ORSA J. Comp. 1(1989)190–206.

[3] F. Glover, Candidate lists and tabu search, Preprint, CAAI, University of Colorado (1989).

[4] A. Hertz and D. de Werra, The tabu search metaheuristic: How we used it, Ann. Math. AI 1(1990)111–121.

[5] M. Lee, Least cost network topology design for a new service using the tabu search, in: *Heuristics for Combinatorial Optimization*, Section 6 (1989), pp. 1–18.

[6] M. Malek, M. Guruswamy, H. Owens and M. Pandya, Serial and parallel search techniques for the traveling salesman problem, Ann. Oper. Res. 21(1989)59–84.

[7] S.H. Parrish, T. Cox, W. Keuhner and Y. Qui, Planning for least cost expansion of communications network topology, US West Advanced Technologies Science and Technology Technical Report.

[8] J. Ryan (ed.), Final report of mathematics clinic, in: *Heuristics for Combinatorial Optimization* (1989).

[9] J. Skorin-Kapov, Tabu search applied to the quadratic assignment problem, Research Report HAR-89-001, W.A. Harriman School for Management and Policy, SUNY at Stony Brook, NY (1989), to appear in ORSA J. Comp.