December 1998

Telecommunications Network Design: A Genetic Algorithm Approach

Hsinghua Chou
Iowa State University Ames

Chao-Hsien Chu
Iowa State University Ames

G. Premkumar
Iowa State University Ames

Telcommunications Network Design: A Genetic Algorithm Approach

Hsinghua Chou
G. Premkumar
Chao-Hsien Chu
College of Business
Iowa State University, Ames

Introduction

The network design to support a digital data service (DDS) is a major design issue for telecommunications organizations. Typically, the DDS network consists of three components - hubs, end-offices, and customer locations. The hubs are the primary nodes that form the backbone infrastructure. The customers are connected by leased lines to the local end-office and the end-offices are in turn connected to the hubs. Typically, the customers are always connected to one end-office and each end-office is connected to one hub or node, thereby creating a star topology in the local access network. There are costs involved in setting up and operating the hub, the links connecting the hubs, links from the end-office to hub, and links from the customer to the end-office. The network designer is primarily interested in designing a network infrastructure that meets the customers' requirements at minimum cost.

The design problem discussed above is a classic Steiner tree star (STS) problem of finding the minimum cost tree connecting a set of nodes, using Steiner nodes, where each target node (end-offices) is connected to only one active Steiner node in a star topology. The Steiner tree problem can be formally stated as: There are *m target nodes*, which are interconnected through *n Steiner nodes*. Each Steiner node *j* that is used to connect at least one target node or other Steiner nodes is called an active Steiner node and will incur a fixed cost $b_{j..}$ Each target node *i* must be connected to exactly one active Steiner node, incurring a connection cost $C_{ij\,.}$ In addition, two distinct Steiner nodes *j* and *k* that are directly connected incur a connection cost d_{jk}. We need to find the minimum cost tree that spans all target nodes through selected active Steiner nodes. For brevity sake, the mathematical formulation of the model is not presented in this paper.

The STS problem is a classic combinatorial optimization problem. The number of constraints increases exponentially as the problem size increases making it infeasible to use mathematical modeling tools for medium to large size problems. There are two innovative approaches to solve the STS problem - Tabu search heuristic and genetic algorithms.

Tabu Search

Tabu search is a meta-heuristic search to overcome 'local optimality entrapment' in optimization problems. A Tabu list is formed that maintains the most recent solutions, and in each iteration of the optimization process the solutions are checked against the Tabu list. A solution on the list is not chosen for the next iteration, thereby preventing cycling in the local solution space. In each iteration, the steepest-descent solution that does not violate Tabu condition is chosen. More details about the method are available in Glover, Kelly, and Laguna (1995). Tabu Search heuristic has been used in a variety of network design problems. The Tabu search algorithm starts with an initial feasible solution. Typically, in a backbone network design problem, a 'greedy' algorithm such as Prim's algorithm, is used to develop an initial minimal spanning tree solution. Then a neighborhood space is searched for a series of exchange. Each exchange is compared with the Tabu list to check if this solution exists. Intensification and diversification strategies are used to improve the search. More details on the Tabu search heuristic is available in Xu, Chiu, and Glover (1996).

Genetic Algorithms

Genetic Algorithm (GA), introduced by John Holland (1975), refers to a class of adaptive search procedures based on principles derived from natural evolution and genetics. The "survival of the fittest" is the basis for this class of algorithms. Recently, GA has been used in a wide variety of network design problems. The key components of GA, described below, are: problem representation, population initialization, selection, evaluation using a fitness function, and reproduction using crossover and mutation.

The *problem is represented* using chromosomes, which are finite length strings that represent important characteristics of the problem. Most GA implementations use binary strings for encoding. After carefully analyzing the traits of the STS problem, we infer that the key determinant to solving the STS problem lies in determining of hub locations. Since the end-offices have a fixed location, we can ignore the encoding of the end-offices. In our binary scheme each gene represents a Steiner node (hub). Each gene takes a value of either 0 or 1, where 1 indicates that it is an active hub and 0 indicates that it is inactive. The length of the encoding string is equal to the number of hubs.

The *population initialization* is done using a random initialization strategy. In our case, a random number generator was used to generate 0 or 1 and then assigned to each gene until all the hubs had a number.

The *fitness function* evaluates the total cost of network solution. The objective of the Steiner-Tree-Star problem is to connect the hubs to the target nodes at the minimum cost. Since the coding strings contain the active/inactive status of the hub, the problem can be transformed into a minimal spanning tree problem (MST), where all the active nodes in the encoding need to be connected at the minimum cost. Hence, 'prim' algorithm, which is extensively employed to generate a MST, is used to obtain the initial solution.

The Prim's MST algorithm can be illustrated as follows

S0. G is the given non-trivial n-vertex weighted connected graph.

S1. Set $i = 1$ and $E =$ Æ. Select any vertex, say v, of G and set $V_1 = \{v\}$.

S2. Select an edge $e_i = (p, q)$ of minimum weight such that e_i has exactly one end vertex, say p_i. Define $V_{i+1} = V_i \cup \{q\}$, $E_i = E_{i-1} \cup e_i$, and $T_i = E_{i-1} \cup e_i$.

S3. If $i < -1$, set $i = i + 1$ and return to S2. Otherwise, let T_{min} - T_{n-1} and Halt.

The objective function is calculated by aggregating the connection, bridging and fixed node cost. The least cost of connecting each target node (end-office) with a node on the backbone network is determined by selecting the closest (cheapest) active node. A fitness value is assigned to each chromosome based on the total cost of backbone and local access network. The fitness function to evaluate the chromosomes compares each chromosome with the highest value in the population. Chromosomes with higher fitness values are retained for next generation.

The *selection* decision identifies the strategy for creating the mating pool for the next generation. There are two possible approaches to selection of the mating pool - use only the offspring, or use a mixture of parent and offspring population. In the second approach the parent and offspring populations are mixed and the "fittest" from this pool are selected. Researchers prefer selection from an enlarged pool since it reduces possibility of duplicate chromosomes. There are two methods in the enlarged pool option - mix them and choose the required number for the new population ($\mu+\lambda$), or choose a fixed number from offsprings and the remaining from the parent population (μ, λ) to create the new population. In our study we used the enlarged ($\mu+\lambda$) selection procedure.

The *completion criterion* determines when the iterative process is complete. The completion decision can be based on three criteria - number of total generations, execution time, and fitness convergence. In this study we use fitness convergence as our criteria. It stops the evolution process when all the chromosomes in the population have the same fitness value.

The *reproduction operators, crossover and mutation*, are used to generate offspring from the parent population. In crossover the genes of the parent are interchanged in a certain fashion to create two new offspring with very different characteristics. There are different variations of the crossover operator - one-point, two-point, and uniform crossover. We used uniform crossover in our study. *Uniform crossover* starts from generating, randomly, a set of positions called masks, within the length of the chromosome. Then a pair of chromosome exchange their genes between each other based on the generated positions.

Mutation, unlike crossover, occurs within the chromosome rather than across a pair of chromosomes. Mutation randomly flips some bits in the chromosome. There are two mutation methods - insert and exchange mutation. *Insert mutation* randomly generates two positions in a given chromosome. The algorithm inserts the gene from the first position in the second position and shifts all the genes on the right by one position. *Exchange mutation* randomly selects two positions in a given chromosome and exchanges both genes. The remaining genes are kept intact. We used exchange mutation in our experiment.

Experimental Design

The effectiveness of GA and Tabu search for different networks was evaluated using an experiment. The data set was generated on a grid graph. The procedure randomly select m nodes as target nodes from a s x s grid graph. The remaining s^2 - m grid nodes are considered as Steiner nodes. The setup cost for each Steiner node is randomly generated from the interval [10,1000]. The data sets were generated ranging from a small size problem (10 x10 nodes) to a large size problem (300 x 300 nodes). The experiments were performed on a Digital Alpha Station (255/300 EV4.5) 300 MHz server with 64 Mbytes memory.

Results

The results of the experiment are shown in Table 1. The solution quality (cost) and computation time (CPU-secs) are shown for the Tabu and the GA approach. The values in Table 1 indicate that in terms of solution quality both the methods seem to obtain very similar values most of the time. While Tabu search has better solution in four data sets and GA has better solution in 3 data sets. However, the differences in values are minimal. Paired *t*-tests were performed. The results of paired t-test to determine if the differences were significant indicate that they are not statistically significant In terms of computation time GA search generates solutions faster than Tabu search in almost all data sets. The results of t-tests indicate that the differences are significant at 0.001.

Table 1. The Comparison of Tabu Search (iteration = 30000) and GA Results

Data Set (*M*X*N*)	Genetic Algorithms		Tabu Search	
	Cost	CPU (sec)	Cost	CPU (sec)
10X10	4155	0	4155	2
20X20	7686	0	7686	5
30X30	8755	1	8755	10
40X40	11432	3	11432	13
50X50	13166	5	13166	17
60X60	14649	7	14649	33
70X70	19142	12	19142	42
80X80	19280	20	19280	60
90X90	21292	33	21292	120
100X100	16166	40	16166	180
150X100	19359	134	19359	360
200X100	22948*	243	25102	600
125X125	16307	180	16307	540
175X125	21046	300	21046	900
225X125	26223	361	26213*	1380
150X150	19329	360	19329	1320
200X150	24358	377	24358	1920
250X150	28248	772	28248	2880
175X175	20907	760	20907	2700
225X175	25003	903	25003	4020
275X175	27672	835	27672	4440
200X200	22892	1189	22876*	3900
250X200	26122	1179	26122	5520
300X200	29879	1557	29879	7440
250X250	25566*	2330	25573	6940
300X250	29310	3120	29310	8520
350X250	32290	2580	32290	6900
100X300	13120	1654	13120	2510
200X300	21238	3060	21238	3600
300X300	28732	4500	28728*	6120

* *N*: Steiner nodes (hubs), *M*: Target nodes (end offices)
* indicates that the method performs better than the other

The results indicate that the two methods generate solutions that are comparable. Prior researches in this area also confirm that Tabu and GA often generate solutions that are optimal or very near optimal. Hence, significant improvement in solution quality is not feasible. The GA approach consistently provided faster solutions than the Tabu approach, which is consistent with prior research that found computation time for basic Tabu search to be significantly higher than for other heuristic methods. However, the computation time can be reduced by integrating additional heuristics in Tabu search and reducing the number of iterations.

References

References be provided on request.