Designing B-ISDN Network Topologies Using The Genetic Algorithm

Zhigang Qin[1†], Felix F. Wu[2‡], and Nathan Law[‡]

[†]Optimal Networks
1057 E. Meadow Circle, Palo Alto, CA 94303
qin@optimal.com

[‡]Department of Electrical Engineering
and Computer Sciences,
University of California, Berkeley, CA 94720.
{ffwu,hoangla}@eecs.Berkeley.EDU

Abstract

In this paper, the topology design of B-ISDN networks is addressed. We model the topological planning as a nonlinear mixed-integer programming problem. The genetic algorithm, an effective optimization method, is applied to this problem. Since the randomness of the genetic algorithm cannot guarantee the biconnectivity requirement in the topologies generated by the genetic algorithm, we propose an algorithm to make all topologies at least biconnected while increasing the overall cost of the topologies the least. The result for a 20-node test case is presented in the paper and it is shown that the algorithm we propose has a very good convergence property.

1. Introduction

B-ISDN (Broadband Integrated-Services Digital Network) is expected to be the main carrier of next generation communication networks and the Asynchronous Transfer Mode (ATM) will serve as the basic transmission technology for B-ISDN. The B-ISDN technology and the rapidly increasing information technologies will lead to rapid increasing in traffic load and frequent shift in traffic pattern. The changing traffic pattern and the new technologies used in ATM networks make the topological design of ATM networks a new problem. Moreover, most researches that dealt with the topological design were either for data packet networks or for circuit switching networks, and most technical papers about topological design were published during the late 1960s and early 1970s [1]. To the knowledge of the authors, there is no technical paper about the topology planning for the ATM networks. Therefore, we extend our previous work on network planning of packet networks [2] to ATM networks.

Because of the rapid expansion in both circuit-switched and data packet networks, topological planning has long been an active research topic. However, there is still no closed-form algorithms for optimizing a network topology for circuit-switched or data applications. The designer must employ a combination of heuristic algorithms and analyses in an interactive manner until the satisfactory solution is obtained. In [1], network planning methods which rely on human interventions are summarized. This method emphasizes the expert experience and tries to find a satisfactory solution which uses a specific type of topology like star, ring, *etc*. Heuristic algorithms which are not based on a specific topology were also proposed. Perturbation techniques are the main approaches. The most important perturbation techniques are the branch exchange method, cut-saturation method, and concave branch elimination method [10]. Of all the three types of heuristic algorithms, cut-saturation algorithms gives the better results and is computationally more efficient than the other two. In practice, these algorithms are usually combined with other heuristic algorithms or some variations are used to improve result and computational efficiency. The main drawback of these methods is the likelihood to be trapped in a local optimum at an early stage. And there are indeed many local optima for the topology design problem [11]. Algorithms based on integer and nonlinear programming techniques were also proposed [3][4]. The problem with this kind of technique is that it is difficult to deal with large networks and simplifications or assumptions have to be made to make them solvable. Thus, they lack flexibility and are even not realistic in practice.

Because of the drawbacks in the existing algorithms, we propose to use a different optimization technique, genetic algorithm, which can take the advantage of the layered architecture of ATM networks and may find the global optimum, to solve the problem of ATM network planning. In subsequent sections of this paper, the details of this approach is discussed. Section 2 presents the model for the ATM network planning. Section 3 describes the algorithm for solving the problem. In section 4, results are presented for a 20-node ATM network. Conclusions and suggestions for future work are given in Section 5.

2. Mathematical Model for ATM Network Planning

ATM is a layered architecture. Three layers have been defined to implement the features of ATM. The top layer is the ATM Adaptation Layer (AAL). The second layer is

1. He was with Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.
2. Currently on leave with the University of Hong Kong.

the ATM layer. The bottom layer is the physical layer. The ATM connection is set up in the following way: when information needs to be communicated, the sender negotiates a "requested path" with the network for a connection to the destination. When setting up this connection, the sender specifies the type, burstiness, other attributes of the call, and the requirements of the end-to-end quality of service. The network checks its own resources to see whether such a quality of specification can be guaranteed. If yes, a virtual channel connection will be set up, otherwise, the request will be rejected. To accurately model those characteristics of ATM networks, we propose to use a layered framework for network planning and management. In the following paragraphs, we describe the framework in detail.

The planning and management task is divided into four levels. The first level is the physical network design, which is performed to accommodate traffic over a relatively long period, *e.g.*, one year or a few years. The traffic requirements at this level are merely an estimate. Thus, complete characterization of the traffic dynamics complicates the problem and is unnecessary. The task at this level is to design the network topology and assign physical link capacities based on the long-term traffic requirements. The objective of topology design is to minimize the total cost. The second level is the design of the logical network, *i.e.*, the reconfiguration of the network, given the physical network and a more detailed traffic description. The task at this level is to determine the virtual paths and assign them associated bandwidths. The third level is the call control level where admission control takes place. At this level, a more detailed description of the incoming call will be given and the admission control functions decide whether to accept the call based on the resources available at the moment. The fourth level is the cell level. The main task here is to police the traffic so that users will not send more traffic than agreed upon by the network and the users.

The layered model simplifies the task of the topology planning, since the QOS requirements are moved down to the lower levels. According to the layered model, the topological design problem can be stated as follows:

Given:

- node locations
- traffic requirements
- link cost matrix (fixed and variable costs)
- nodal costs

Objective:

Select the links and their capacities to minimize the total cost.

Subject to:

- traffic flow constraints
- reliability constraints
- traffic requirements

A mathematical formulation for the topological design problem is given in following:

Variable definition:

sd: source destination pair

D: set of destination hosts

N: set of all nodes

S: set of source hosts

$$x_{ij}^{sd} = \begin{cases} 0 & \text{pair } sd \text{ does not choose link } ij \\ 1 & \text{pair } sd \text{ chooses link } ij \end{cases}$$

y_{ij}^{sd} : bandwidth assigned to link ij for pair sd;

t^{sd} : the average traffic requirements for pair sd;

c_{ij} : variable cost for link ij (per unit bandwidth);

c_{ij}^{f} : fixed cost for link $ij;$

c_i^n : nodal cost for node i (per unit capacity);

Mathematical formulation:

$$\min \sum_i \sum_j \sum_{sd} (c_{ij} y_{ij}^{sd} + c_i^n y_{ij}^{sd} + x_{ij}^{sd} c_{ij}^{f}) \qquad (1)$$

S.T.

$$\sum_j y_{sj}^{sd} \geq t^{sd} \qquad s \in S, d \in D \qquad (2)$$

$$\sum_i y_{id}^{sd} \geq t^{sd} \qquad s \in S, d \in D \qquad (3)$$

$$\sum_j y_{ij}^{sd} = \sum_j y_{ji}^{sd} \qquad i \in N, i \notin S, i \notin D, s \in S, d \in D \qquad (4)$$

$$y_{ij}^{sd} \geq 0 \qquad i, j \in N, s \in S, d \in D \qquad (5)$$

$$x_{ij}^{sd} = 0 \ \text{ if } \ y_{ij}^{sd} = 0 \qquad i, j \in N; \quad s \in S; d \in D \qquad (6)$$

The objective function (1) is to minimize the total cost, which consists of link cost and nodal cost. The link cost consists of two parts: the fixed cost and the variable cost. The fixed cost of a link does not depend on the capacity of the link, and only depends upon the decision of whether to set up a link there. This could be interpreted as the infrastructure cost of building a link between two nodes. Since the variable x_{ij}^{sd} can only take either 1 or 0 values, the summation of x_{ij}^{sd} will either be 1 or 0, even if multiple routes choose the same link. Thus the fixed cost will not be counted multiple times. The variable cost of a link is proportional to the capacity of the link. So, the link cost is modeled as a piecewise linear function. It should be pointed out that the algorithm we propose has the ability to handle any functions, even functions which do not have close-form formulations. Equations (2) and (3) specify that all traffic requirements must be satisfied. Since we are in the planning stage, the available link capacity may not be the exact amount of the traffic requirement. Therefore, the link capacity could be greater than the traffic requirement. Equation (4) specifies that the traffic flow is conservative, *i.e.*, for any intermediate node, the traffic that goes in the node should equal to the traffic that comes out of the node. Equation (5) states that the link bandwidths cannot be negative. Equation (6) associates the decision variable x_{ij}^{sd} with the bandwidth variable y_{ij}^{sd}.

The above formulation is a non-linear mixed-integer programming problem, which is very difficult to solve. In following section, we propose to use the genetic algorithm to solve this problem.

3. Algorithm

The characteristic of the topological design problem formulated above is that the number of constraint equations quickly becomes unmanageable for even a small problem, if a conventional solving technique is used. If a heuristic approach, such as the perturbation algorithms, is used to solve the problem, only the local optima can be found. Thus, a different, stochastic optimization technique, simulated annealing algorithm, is proposed for topology planning of packet-switched networks in [8] and is shown to be successful. The advantage of a stochastic search algorithm is the likelihood of finding the global optimum and the applicability to various problems. Therefore, in this research, we propose to apply a stochastic algorithm, genetic algorithms, to study the problem, because genetic algorithms have a better chance than simulated annealing of finding the globally optimal solution [6]. The disadvantage of genetic algorithms is that they may be computationally more intensive than simulated annealing.

The genetic algorithm consists of three main procedures: the determination of the initial solution pool, the selection of parents, and the production of offspring. The generation of initial solution pool is to create a population of chromosomes, which represent feasible solutions to the original problem. A good initial solution pool should be randomly selected. After the initial solution pool is selected, some solutions are selected based on the evaluation of the each individual solution (fitness) and the outcome of random choices as the parents of next offsprings. The next step is the production of offspring. The production of new offspring typically undergoes crossover and mutation. Crossover combines and mixes different individuals being selected to form new ones. Mutation is performed on each individual which changes each gene of the individual with a small probability. After these procedures, we get a new population. The new population will display patterns of behavior that are more like those of the successful individuals of the previous generation, and less like those of the unsuccessful ones. With each new generation, the individuals with relatively better values will be more likely to pass on to next generations, while the relatively unsuccessful individuals will be less likely to pass on [7]. The solutions are improved by going through a large number of generations. In 1994, Rudolph Guenter proved that the simple genetic algorithm converges to the global optimum [5].

The difficulty in developing a genetic algorithm to solve a particular optimization problem lies in the necessity of developing appropriate representation and encoding scheme for the solution space. The performance of a genetic algorithm heavily depends on solution representation, encoding scheme, and selection of genetic operators.

For the ATM network planning problem, we propose to obtain an initial solution pool by the following procedure: 1) specify a value k which is the degree of the network that we are going to generate. Start from node 1 to node n (the total number of nodes), for every node x, determine the number of links l which are incident upon this node. Assume l is less than k, determine the k-l neighboring nodes $y_1, \dots y_{k-l}$ which have the least link costs (variable cost) with node x and do not have a connection with x. Make connections between x and $y_1, \dots y_{k-l}$. When we finish checking all the nodes, a k-degree network topology is obtained. Thus m initial network topologies, which make up the initial solution pool, can be obtained by giving m different values to k. 2) check the connectivity of these network topologies. If any of the topologies is not a connected graph, we add the least-cost link which connects the disjoint components of the topology to make it connected.

After the initial topologies are generated, the next step is to assign capacities to links. This is essentially a routing problem. In our study, the shortest path routing in terms of the link cost is used. We proceed as follows: for all source-destination pairs, one at a time, we find the shortest path between the source and the destination, and increase the capacities of all the links along the path with the traffic requirement of this source-destination pair. After the

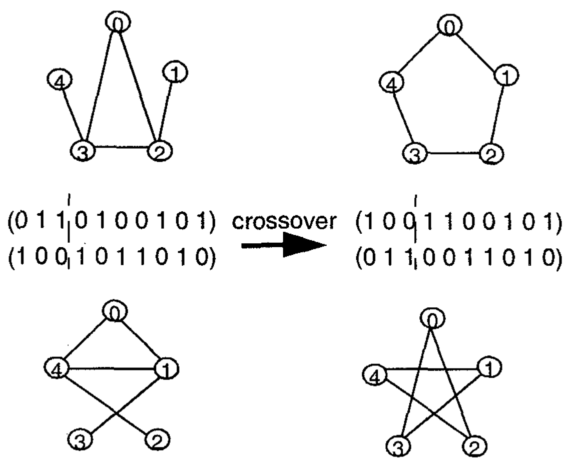

Fig. 1. Illustration of Crossover Operator

capacity assignment is finished, an initial solution pool is established.

Next we evaluate the solutions generated and select candidates for producing offspring. The following fitness function is proposed for the evaluation:

$$f_i = \frac{1/c_i}{\sum_j 1/c_j} \tag{7}$$

where c_i is the cost of *i-th* solution in the solution pool. The parents are selected according to the fitness of each individual, *i.e.*, the probability of each individual being chosen is the fitness value. To make the algorithm converge, we force the selection to include the best solution so far.

The offspring is produced by crossover and mutation. For crossover and mutation operations, a representation of the topology and an encoding scheme are needed. We represent a topology with a *0-1* matrix *topo*. The matrix element *topo(i,j)* equals 1 if there is a link between node *i* and node *j*, and equals 0 if there is no link between them. Because the matrix is symmetric, we only need the upper-triangle elements to represent the topology (diagonal elements excluded). Therefore, the topology can be represented by a binary triangle matrix. Since the crossover and mutation operators need a binary string representation, we combine the upper-triangle elements to form a binary string with $n(n-1)/2$ elements, which will undergo the crossover and the mutation. The crossover operator we adopted interchanges the elements of the two topology strings up to a point which is randomly determined. This process is shown in Fig. 1. After the crossover operations, the mutation operator is applied to all newly generated topology strings. The mutation operator randomly selects a small portion of the elements in the selected candidates and changes the value of the elements to the reverse value, for example, from 1 to 0 or from 0 to 1. In other words, mutation operator takes off or adds on some links randomly from some offspring topologies. The percentage of links which go under mutation operation should be less than 3%. In fact, crossover and mutation are very similar to perturbation methods in the sense that they all search for the best solution by adding and removing links. The genetic algorithm searches the better solution by giving better chance to good solutions to be selected and tries to combine the good parts of two solutions to form a even better solutions through crossover operations. The mutation operations try to lead the search out the local optima or give a new start point for searching. The advantage of the genetic algorithm over the perturbation methods is that it works on a pool of solutions instead of one, and the adding or removing of links is not limited to those operations that will improve the objective value, so the algorithm has a better chance of escaping local minima [6].

After the crossover and the mutation operations, we assign capacities to the links of all topologies using the same shortest path routing algorithm as the one used for generating the initial solution pool. A new generation of solutions is produced when the capacity assignment is finished.

One important point should be pointed out is that a disconnected topology may be produced by the crossover and the mutation. Adding back links is crucial to the algorithm for maintaining feasibility of offsprings and reducing their costs. The proposed approach is to add back the minimum cost links which connects the disjoint components. The procedure is as follows: assume that a disconnected topology is generated by the crossover and the mutation operations. First, we select a source-destination pair which has a source in one subnetwork and a destination in the other subnetwork. Then we use Dijkstra's algorithm [9] to generate two minimum-cost spanning trees rooted from the source and the destination of this source-destination pair for both disconnected subnetworks. Then we compare the costs of all links that connect the leaf nodes of these two disjoint components, and add the link of the least cost to the topology. This process is repeated for all other source-destination pairs which have a source in one subnetwork and a destination in the other subnetwork without considering the newly added links (otherwise, the network is already connected and there is no need to add a link to the topology). After this operation, the topology will become connected again.

The above algorithm does not consider the reliability constraint. For network in which the reliability is important, biconnectivity (2-connectivity) or n-connectivity constraints are required. In practice, biconnectivity is normally good enough for reliability concern. Moreover, more connectivity implies more redundant resources. Thus, only biconnectivity constraint has been considered in our algorithm. Before we explain the algorithm of converting non-biconnected graph to a biconnected graph, we introduce a new concept, a lemma and a theorem.

Definition. A biconnected component is called peripheral if it has only one articulation point.

Lemma. Let G be a graph, G1 and G2 be two peripheral biconnected components of the graph G, and G3 be the smallest subgraph of G which contains G1 and G2. If a link is added between any nodes of these two components except for the articulation points of G3, then resulting subgraph G3 is biconnected. For all such links, the link which has the minimum cost is called the bridge link.

Proof. Before the link is added, the only node whose removal will split the graph is the articulation point. After this link is added, this is no longer true.

Theorem. Let *n* be the number of peripheral biconnected components of a non-biconnected graph. Then the minimum number of links needed to make it biconnected is *n-1*. The minimum cost to make it biconnected is the cost of the *n-1* bridge links.

Proof. By the lemma above, we need at least one link to make a graph with two peripheral biconnected components biconnected and the bridge link is the least cost link of all such links. Adding such a link also reduces the number of peripheral biconnected components by 1. Thus, for a graph with *n* peripheral biconnected components, by induction, we need at least *n-1* links to make the graph biconnected and the minimum cost is the cost of the *n-1* bridge links.

Based on the discussion above, we propose an algorithm for ATM network planning with biconnectivity constraint. The block diagram of the algorithm is shown in Fig. 2. In the algorithm, the biconnectivity constraint is checked after a network topology is generated. If the topology is not biconnected, a bridge link is added to the topology to make it biconnected. The link capacities are assigned according to the shortest path routing algorithm which increases the capacities of the links along the routes by the amount of traffic requirements. For the links which are not among the shortest paths, there are two ways to assign the capacities. The first way is to assign the minimum link capacity given by users. The second way is to assign the minimum alternative path capacity. The minimum alternative path capacity is defined as follows: assume that concatenated links *i*,..., *j* are not among any of

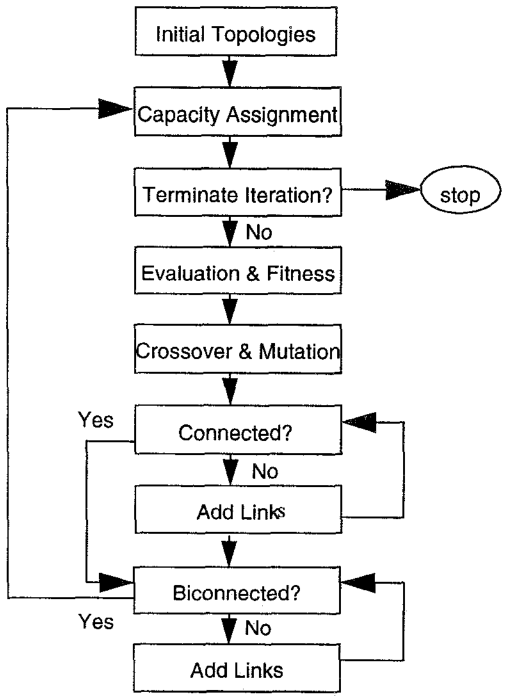

Fig. 2. The control flow of topology design with biconnectivity constraint

the shortest paths and nodes *i* and *j* are the terminating nodes of these links. The minimum alternative path capacity is the minimum path capacity of all alternative routes from *i* to *j* (since the topology is biconnected, the alternative route always exists). The default way is the first way.

In the following section, the results of applying this algorithm to a 20-node test case are presented.

4. Results

The algorithm has been applied to different network planning cases. The result of a 20-node network design problem is presented here. The traffic among the nodes is assumed to be uniformly distributed among all nodes. The link costs are obtained from [10]. The node cost is assumed to be proportional to the capacity of the node. The number of solutions in the solution pool is set to be 20. To test the performance of the algorithm under different settings, we alter the value of the crossover probability and the mutation probability. The costs of the best solutions at each iteration for different settings are shown in Fig. 3. From the figure we can see that the probability of mutation has a large effect on the performance. A larger probability of mutation significantly alters the topology and results in a slower rate of convergence to the optimum. The probability of crossover also affects the rate of convergence. In this case, both the highest and the lowest crossover probabilities result in a slower rate of convergence. For all circumstances, as the costs of the best topol-

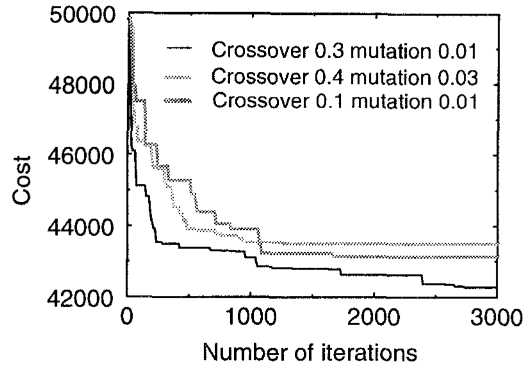

Fig. 3. Results of network with 20 nodes

ogies at each iteration approach to the minimum solution, the convergence speeds slow down. It is also shown that a 15% reduction in the cost has reached after 3000 of iterations when crossover probability and the mutation probability are set to be 0.3 and 0.01 respectively. Other choices of the crossover and the mutation probabilities perform slightly worse. Table 1 summarizes the results for the 20-node case.

This algorithm can also be applied to the case which requires the planning to include the existing networks with minor changes. To force all the solutions to contain the existing networks, we first make all the initial solutions (topologies) to contain the existing networks (subgraphs). Then, we change the mutation operator such that the mutation operations do not alter these subgraphs. Since all the solutions in the solution pool contain the subgraphs, the crossover operations will keep the subgraphs in the resulting topologies. Thus the main changes are the process of generating initial solutions and the mutation operator.

Table 1: Network Design Results

Iteration	Cost reduced	Cost added for bi-connect.	Crossover prob.	Mutation prob.
3000	15.1%	0.6%	0.3	0.01
3000	13.1%	0.8%	0.4	0.03
3000	13.9%	0.8%	0.1	0.01

5. Conclusions and Future Work

We propose a layer framework for ATM network design. The layer approach makes the design task more manageable. Although topological design is still a large scale, non-linear, mixed-integer programming problem, we are able to solve the problem using a genetic algorithm. The algorithm is tested with a few test runs. From the test results, it is shown that the genetic algorithm we develop is effective.

The possible future work for ATM network planning includes the expansion of the algorithm to more complicated cases, for example, the case that the node locations are not given. In general, it is very difficult to solve this problem; however, for a special case that the possible candidates for the node locations are known, the genetic algorithm may still be applied, though some changes have to be made. The new requirement for this case is that the candidate nodes for one location are exclusive, *i.e.* only one of them should be selected into the solution. Thus, we can change the crossover and mutation operator such that for all node locations, only one of their candidates is selected into the resulting topologies.

6. Reference

[1] R. L. Sharma, "Network Topology Optimization", *New York: Van Nostrand Reinhold, 1990.*

[2] Shau-Ming Lun, Felix Wu, Ning Xiao, and Pravin Varaiya, "NetPlan: An Integrated Network Planning Environment", *State of the Art in Performance Modeling and Simulation: Computer and Communication Networks, Vol.1, Kluwer Academic Publisher,* 1993.

[3] B. Gavish, "A General Model for the Topological Design of Computer Network," *Proc. of GLOBECOM'86,* pp. 1584-1588, 1986.

[4] A. Gersht and T. Weihmayer, "Joint Optimization of Data Network Design and Facility Selection," *IEEE Journal on Selected Areas in Communications,* vol. 8., No. 9, pp. 1667-1681, Dec. 1990.

[5] D. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," *Mass. Addison-Wesley Pub. Co.,* c1989.

[6] Z. Michalewicz, "Genetic algorithms + data structures = evolution programs," *Berlin; New York: Springer-Verlag,* 1994

[7] A. Chuang, "An Extendible Framework for Genetic Algorithms in the Ptolemy Environment," *MS report,* University of California at Berkeley, 1995.

[8] Ning Xiao, "Capacity planning and Topology Optimization of Corporate Communication Networks," *Ph.D thesis,* University of California at Berkeley, 1993.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, "Introduction to Algorithms," Cambridge, *Mass.: MIT Press*; New York: Mcguire-Hill, 1989.

[10] Ricardo F. Garzia and Mario R. Garzia, "Network Modeling, Simulation, and Analysis," *New York: M. Dekker Inc.,* 1990.

[11] D. Bertsekas and R. Gallager, "Data networks," 2nd ed. *Englewood Cliffs, N.J.: Prentice Hall,* c1992.