Optimal Topological Design of Communication Networks Using Generational/Steady-State and Struggle Genetic Algorithms

HAMEIDA SAYOUD, KENZO TAKAHASHI, BENOIT VAILLANT, NICOLAS TAILLADE
Center of High Speed Broadband Networking, Faculty of Engineering
Multimedia University
63100, Cyberjaya
MALAYSIA

***Abstract**:* This paper presents three variants of the simple Genetic Algorithm (GA) with specialized encoding, initialization and constraints handling mechanisms to optimize the design of topologies and capacity assignment (TDCA) problem of broadband communication network. This NP-Complete problem is often a highly constrained optimization problem that is better solved using GAs combined with specialized, problem specific genetic operators. A network of 20 nodes is used to test the developed algorithm. Improved results, both in network performance and computation speed, are obtained when comparing with existing heuristic approaches. Similar methodology can be applied for the design of other communication networks.

***Key-Words**:* Genetic Algorithms, Asynchronous Transfer Mode, Topology Design, Capacity Assignment.

1 Introduction

In broadband ATM networks, the design of an optimal minimum cost network of N nodes can be considered to consist of two sub-problems; (1) topology identification and, (2) capacity assignment of the chosen links. The presence of multiple routes between each origin and destination nodes requires the solution of complicated routing and capacity allocation problems. The installation of the physical link is associated with a number of cost figures. The geographical layout, types and capacity of the links determine the cost of setting up a network. After defining the optimal network topology, the capacity assignment has to be so as to ensure that the predicted traffic on the network can be accommodated. In addition to the cost economics, routing and capacity concerns, the network designer has to guarantee that the designed network will meet some performance characteristics. Studies of network design generally seek to optimize average network delay, throughput or reliability [1][2][3][4][6][8][9][10][11]. Therefore the ATM network design problem presented in this paper can be summarized as an optimization of cost parameters subject to the traffic requirement and performance criteria.

Because of the combinatorial nature of the TDCA problem, it has been classified as NP-Complete [22] problem; that is a problem that belongs to a class of problems for which no polynomial-time algorithms are known which can guarantee exact solution. Due to their complexity, requirement for organized search through the solution space and exponentially growing calculation time, this type of problems are best solved using heuristic algorithms. GA [23][24], a promising heuristic search procedure that applies natural genetic ideas such as natural selection, mutations and survival of the fittest for solving large combinatorial problems has been chosen.

GAs have been applied to a variety of communication network optimization problems with different degree of success. Theses problems range from bandwidth allocation of embedded ATM networks [4][12], routing and wavelength allocation [22], to survivable network design [13]. For the topological design, Kumar *et al.* [13] developed a GA considering diameter, distance, and reliability to design and expand computer networks. Deeter and Smith [10] presented a GA based approach to design network considering all-terminal reliability with alternative link options, allowing edges to be chosen from different components with different costs and reliabilities. Walters and Smith [16] proposed an evolutionary algorithm for the optimal layout design of networks with a tree structure. Dengiz *et al.* [8][9] focused on large backbone communication network design considering all terminal network reliability and used a GA, but appreciably customize it to the all-terminal design problem to give an effective and efficient optimization methodology. Elbaum and Sidi [3] used GAs based on a Huffman tree to solve the topological design of LAN networks. Gen *et al.* [5] proposed a GA for solving bicriteria network design problems considering connection cost and average message delay without considering network reliability. Jong Ryul *et al.* [21] proposed a GA for solving the same problem considering network reliability and using Prüfer number and cluster to represent chromosomes. Pierre and Elgibaoui [7] presented a Tabu-search approach for designing computer network topologies with unreliable components.

To the knowledge of the authors, there have been few published works [1][2] [15] dealing with the solution of the TDCA using genetic or other similar evolutionary algorithms. The purpose of this paper is to present three different GA approaches to the solution of this problem. Section 2 introduces the statement and the mathematical formulation of the problem. Section 3 overviews the three GA variants as well as genetic operators and repair mechanisms. Section 4 analyses computational experience, and compare the performance of the presented approaches with other heuristics methods.

2 Statement of the Problem

Given:

- Node locations
- Traffic requirements
- Fixed and variable link costs
- Nodal costs

Objective:

Select 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 and capacity assignment problem is as follows:

Variable definition:

sd: source destination pair; *D*: set of destination hosts; *N*: set of all nodes; *S*: set of source hosts

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

y_{ij}^{sd} = bandwidth assigned to link *ij* from source destination pair *sd*; t^{sd} = average traffic requirements for source destination pair *sd;* c_{kj} = variable cost of link *kj* per unit bandwidth; c_{kj}^{f} = fixed cost for link *kj;* c_{k}^{n} = nodal cost for node *k*. In mathematical form, the TDCA can be stated as follow:

$$\text{Minimize} \quad \sum_i \sum_j \sum_{sd} \left(c_{ij} y_{ij}^{sd} + c_i^n y_{ij}^{sd} + x_{ij}^{sd} c_{ij}^f \right) \qquad (1)$$

Subject to:

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

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

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

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

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

The objective function (1) is to minimize the total cost, which consists of link cost and nodal cost. The link cost consists of fixed and variable costs. The fixed cost does not depend on the capacity of the link, and depends only upon the decision of whether to set up a link or not. The variable x_{ij}^{sd} will be either 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. Equation (2) and (3) represent constraints on the traffic. Since this problem is solved 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 is conservative. Equation (5) forces the bandwidth to be positive and (6) associates the decision variable x_{ij}^{sd} with the bandwidth variable y_{ij}^{sd}.

3 The Genetic Algorithm

Although much theoretical work on GAs exists, and much more is currently being pursued by the GA community, there does not yet exist a complete theory for GAs that says which GA operators and their parameter values are best. Often when implementing a GA, practitioners rely upon a large body of empirical research that exists in the literature. In some cases this work is theoretically guided; in others it is the result of extensive experiments or specific application case studies.

3.1 GA Variants

One way to classify GAs is by the percentage of the population that is replaced each generation. Two choices are common in the literature, the Generational Replacement GA (GRGA) and the Steady-State GA (SSGA) [23]. A third emerging choice is a class of GAs commonly known as niching methods that allow to allocate and maintain multiple different optimal/suboptimal solutions in a population [17][18][19]

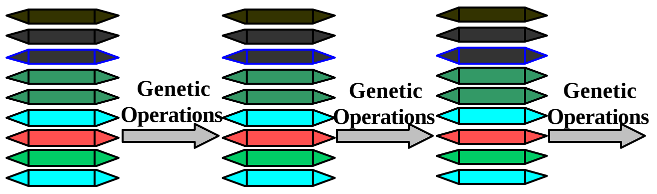


Fig. 1, The simple GA, this algorithm uses non-overlapping populations; the entire population is replaced each generation.

The GRGA uses non-overlapping populations. As depicted in Fig. 1, in each generation, the entire population is replaced with new individuals. Typically, the best strings carrying the important building blocks are carried over from one generation to the next using some sort of elitism so that the algorithm uses what is already regarded as the best that it found.

Since the entire population is being replaced, the crossover plays a very important role in determining the capability of the GRGA to find an optimal solution. If the crossover accurately conveys good genetic materials from the parents to their offspring, then the population will improve. On the contrary, if the best strings are not allocated any reproductive trails or the crossover/mutation operators destroy or alter important bit values in a chromosome, than the population will not improve and the GA will perform no more than a random walk.

The SSGA, is an alternative to the GRGA and uses overlapping populations. In each generation, only few individuals are replaced at a time by the newly generated individuals. This process is illustrated in Fig. 2. The SSGA has a build-in elitism since only the lowest ranked string is deleted. The percentage of population that is replaced, i.e. overlap amount, plays an important role in the convergence behavior of the algorithm. At one extreme, a nearly 100% overlap is obtained by replacing one or two individuals each generation. At the other extreme, the SSGA becomes a simple GRGA if the entire population is replaced i.e. 0% overlap.

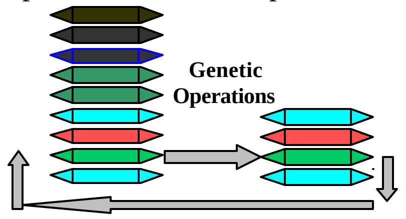


Fig. 2, The Steady-State GA. This algorithm uses overlapping populations; only few individuals are replaced each generation.

In the case of small population size, the SSGA suffers from premature convergence to sub-optimal solutions because the more fit individuals are more likely to be selected and the population quickly converges to a single individual.

As depicted in Fig. 3, the Struggle GA (SGGA) [17][19] is similar to the SSGA except that rather than replacing the worst individual in a population, the SGGA replaces an individual by a new one most similar to it if this new individual has a score better than that of the one to which it is most similar. This requires a distance function to measure the similarities between two individuals in terms of their actual structure (genotype) or their characteristics in the problem space (phenotype).

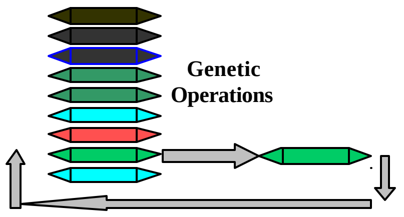


Fig. 3, The Struggle GA. This algorithm replaces the most similar individual; replaces an individual by a new one most similar to it if this new individual has a score better than that of the one to which it is most similar.

If the similarity function is properly defined, the SGGA has the capability to maintain high diversity among solutions. However, like other GAs, performance is tightly coupled to genetic operators. For example, without any robust mechanism to introduce new diversity, the algorithm converges to a single solution. When all individuals in a population have converged to a single genotype, crossover does not contribute to the search since all the information it can process is equal. In this case mutation can be very helpful in exploring (large mutation probability) and exploiting (small mutation probability) the search space. As mutation rate increases more blind diversity will be introduced, thereby preventing the GA from premature convergence. However, the randomness introduced by mutation may not be practical all the time. On the other hand, if there is enough diversity in a population, the crossover operator can provide adequate exploration and exploitation of the search space. If we want to rely on crossover for exploration and exploitation rather on high mutation rates, useful diversity is required [16]. This diversity may be maintained by preserving all promising solutions. Since the population has no prior knowledge of the search space, loosing promising solutions might lead to genetic drift to the easily found promising areas and thus loose important information that may help in locating more new promising areas.

It is important to note here, that once lost, the chance that a mutation will regain that desired information is quite low. The SGGA has the ability to maintain this diversity by minimizing the erroneous replacement of unique and potential promising solutions.

3.2 GA Components

As described in [23], the major components of a GA implementation are as follow:

1. a means of encoding solutions to the problem as chromosomes
2. a means of obtaining an initial population of solutions
3. a function that evaluates the fitness of a solution
4. reproduction operators for the encoded solutions
5. appropriate settings for GA control parameters,

In addition to these five components, problem specific operators are needed, these include:

6. a mechanism to repair the network topology in case of absence of bi-connectivity, and
7. a mechanism to deal with dense graph created by GA operators.

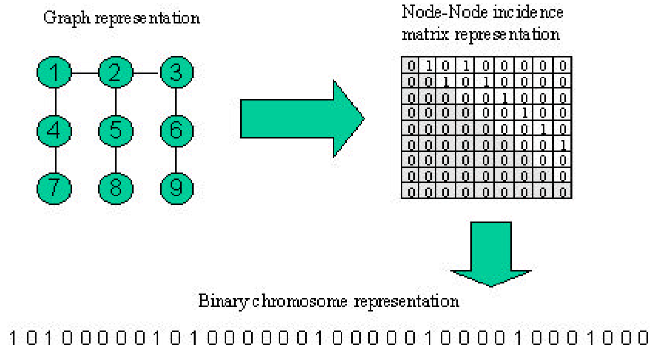
3.2.1 Chromosome Encoding

In order to optimize the topology of the network with a GA approach, a chromosome encoding scheme is needed that can represent a general graph. Any graph can be uniquely represented as a node-node incidence matrix [15]. This provides the simple basis for the chromosome representation used in this paper. Since we are assuming that all links are bi-directional, we will consider the upper triangle portion of the matrix only. This matrix can be converted to a vector without loss of information. If n is the number of nodes in the graph, then:

$$\text{chromosome length} = \frac{n(n-1)}{2}$$

It is important to note that the search space of the problem is proportional to the number of nodes:

$$\text{cardinality of search space} = 2^{n(n-1)/2}$$


Graph representation
1
2
3
4
5
6
7
8
9
Node-Node incidence matrix representation
Binary chromosome representation

0	1	0	1	0	0	0	0	0
0	0	1	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0

Fig. 4, Chromosome Encoding

3.2.2 Initialization

To enhance the efficiency of the search, the initial population consists of networks being highly reliable. A combination of depth-first search algorithm and repair mechanism is used to generate the initial population by the following procedure:

1. Specify a value k, which is the degree of the network that we are interested to generate. Start from node 1 to n (the total number of nodes). For every node x, determine the number of links l which are incident upon this node. If l is less than k, determine the k-l neighboring nodes $y_1, y_2,\ldots,y_{n-l}$ which have the lowest link variable costs with node x and do not have a connection with x. make connections between x and $y_1, y_2,\ldots,y_{n-l}$. Run a check for connection. After checking the connectivities, a k-degree network topology is obtained. Thus m initial topologies which make the initial population 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, add the least-cost link which connects the disjoint components of the topology to make it connected.
3. For all source s and destination d pairs, one at a time, find the shortest path between s-d, increase the capacities of all the links along the path with the traffic requirement of this source-destination pair. An initial solution population will be obtained after finishing the capacity assignment.
4. Evaluate the solution generated and select candidate for matting using Equation (7).

3.2.3 Selection

The selection mechanism allocates reproductive trials to strings on the basis of their fitness. Depending on the type of GA, strings selected from the old generation are either included directly in the new generation or become the parents of new strings created by GA recombination operators. The results reported in this paper are based on the binary tournament selection, where two strings are chosen at random from the population, the more fit string is then allocated a reproductive trial. In order to produce an offspring, two binary tournaments are held, each of which produces one parent string.

3.2.4 Fitness Function

A function is needed to interpret the chromosome and produce an evaluation of the chromosome's fitness. This function must be defined over the set of possible chromosome's and is assumed to return some non-negative value representing the fitness. In our formulation we used the following function:

$$f_i = \frac{1/c_i}{\sum_j 1/c_j} \qquad (7)$$

where c_i is the cost of the i-th solution in the solution population. The parents are selected according to the fitness value.

3.2.5 Crossover

Crossover allows elements from different chromosomes to be combined hoping that good elements will be recombined to produce a chromosome fitter than its parents. We used multi-point crossover, where it is accomplished by selecting two parent solutions, (i.e. interchange the elements of the two topology strings) and randomly taking a component from one parent to form the corresponding components of the offspring.

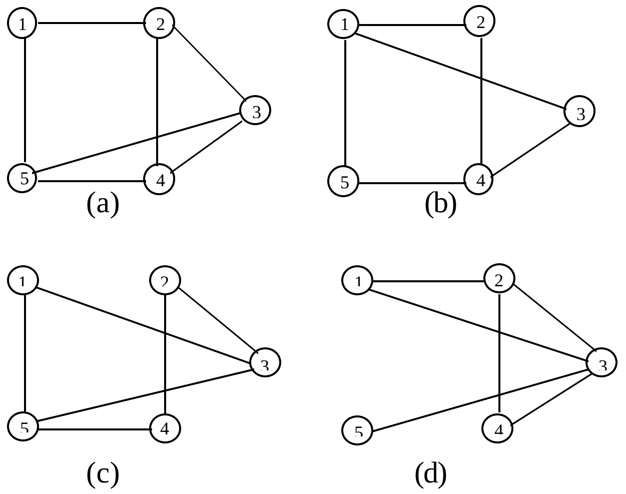


Fig. 5, Two children topologies (c) and (d) created by crossover operator from parent topologies (a) and (b)

3.2.6 Mutation

The purpose of mutation is to create diversity that may not have been present in the initial population by taking off or adding on some links randomly from some offspring topologies. We used a common mutation operator based on a random walk and is applied differently according to node degrees of the network.

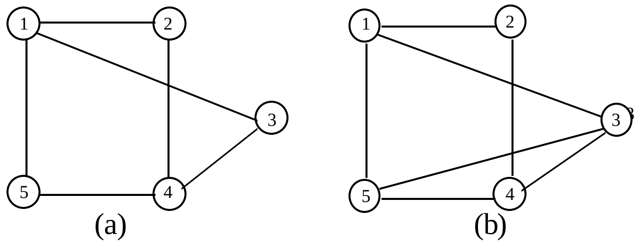


Fig. 6, Mutation of a network of five nodes and six links. (a) Network degree $k \geq 2$ before mutation; (b)Network after mutation where link 3 to 5 has been added.

3.2.7 Repair Mechanism

Since not all chromosomes created randomly or by genetic operators represent a network that meets the defined constraints, a testing and repair process is needed. Connectivity/biconnectivity is easily and cheaply tested using a depth first search. If a network is not connected, random links are added between components until the graph is connected. Each node is checked for compliance to the minimum and maximum degree constraint using the incidence matrix. If a node breaches the minimum degree constraint, one or more extra links must be added. If a node breaches the maximum degree constraint, one or more links must be removed, without destroying connectivity.

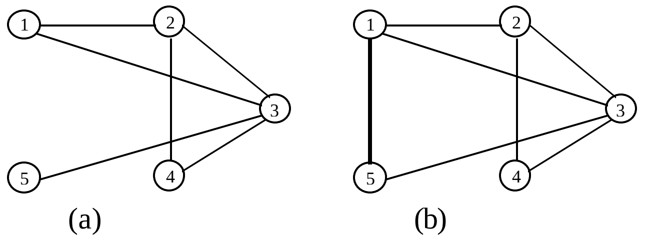


Fig. 7, Topology from Fig. 5.d that has undergone repair for biconnectivity (a). The bold link between 1 and 5 has been added (b).

3.2.8 Pruning

The mutation and repair mechanism operators both add links to the network in order to comply with the imposed biconnectivity constraint. This may create some complications in that the population would drift towards a set of fully connected graphs. We adopted two approaches with closely similar results, either we penalize dense graphs in the objective function or we detect links that are not allocated any traffic and prune them. In this way, the genetic building blocks being recombined in the following generations are the links that are actually used.

3.4 Similarity Measure

The similarity measure is a function used by the SGGA to compares two solutions and indicates the differences between them. The Hamming distance is an appropriate phenotype measurement of similarity for problems where binary encoding matches the phenotypic meaning (network topology). The Hamming distance between two binary strings b_1 and b_2 of length l is defined by the sum of the non-matching bit-values at each position. In general, this distance d_h between two bitstrings can be written as:

$$d_h(b_1, b_2) = \sum_{i=1}^{i} \left| v\{b_1(i)\} xor v\{s_2(i)\} \right| \qquad (8)$$

Where $v\{b_j(i)\}$ is the bit-value of string j at position i.

4 Implementations

Fig. 8, 9 and 10 present the three algorithms used in this paper, in addition to the definition of Section 3.1 the following is a brief explanation of the SSGA algorithm. The other two are not explained here for reasons of space. $P(t)$ is the population of strings at generation t. Each generation, one new string is inserted into the population. The first step is to pick a random string, x_{random}. Next, two parent strings, x_1 and x_2, are elected, and a random number, *r Î [0, 1]*, is generated. If r is less than the crossover probability, p_c, we create two new offspring via crossover and randomly select one of them, x_{new}, to insert in the population. Otherwise, we randomly select one of the two parent strings, make a copy of it, and apply mutation to flip bits in the copy with probability $1/n$. In either cases, the new string is tested to see whether it duplicates a string already in the population. If it does, it undergoes (possibly additional) mutation until it is unique. The least-fit string in the population is deleted, x_{new} is inserted, and the population is reevaluated. The GRGA and SSGA experiments in this paper used a population size of 20, while the SGGA was tested with a population of 50.

```
t ¬ 0
initialize P (t)
evaluate P (t)
foreach generation
        t ← t + 1
        select P (t+1) from P (t)
        recombine P (t+1)
        evaluate P (t+1)
endfor
```

Fig. 8, The GRGA Algorithm

```
t ¬ 0
initialize P (t)
evaluate P (t)
foreach generation
        select($x_1$, $x_2$ ) from P (t)
        if( $r < p_c$ ) then
                $x_{new}$ = crossover($x_1$, $x_2$)
        else
                $x_{new}$ = mutate($x_1$ , $x_2$)
        endif
delete ($x_{worst}$ Î P (t))
while ($x_{new}$ Î P (t))
mutate($x_{new}$)
P (t + 1) / P (t) È $x_{new}$
evaluate P (t + 1)
t ¬ t + 1
endfor
```

Fig. 9, The SSGA Algorithm

```
t ¬ 0
initialize P (t)
evaluate P (t)
foreach generation
    select ($x_1$, $x_2$ ) from P (t)
    $x_{new}$ = crossover ($x_1$, $x_2$)
        $x^*_{new}$ = mutate ($x_1$ , $x_2$)
        Find most similar individual S to $x^*_{new}$
        If $f(x^*_{new}) < f(S)$ Replace S with $x^*_{new}$
        evaluate P (t + 1)
 t ¬ t + 1
 endfor
```

Fig. 10, The SGGA Algorithm

5 Results

The approach presented in this paper is applied to a 20-nodes problem found in [1]. For simplicity and ease of comparison, the traffic is considered uniformly distributed among all nodes and the node costs are considered proportional to the capacities of the links with a unit cost of 3.0.

Iter.	Cross. Prob.	Mut. Prob.	Cost Reduction	Improvement Over [1]	Cost for Biconnect
3000	0.85	0.052	11.50 %	5.98 %	0.92 %
3000	0.65	0.02	11.28 %	5.89 %	0.95 %
3000	0.30	0.015	11.42 %	5.92 %	0.99 %

Table 1. GRGA final results with different settings

Iter.	Cross. Prob.	Mutat. Prob.	Cost Reduction	Improvement Over [1]	Cost for Biconnect.
3000	0.85	0.052	18.78 %	8.79 %	0.99 %
3000	0.95	0.015	18.74 %	8.74 %	0.98 %
3000	0.75	0.052	18.76 %	8.79 %	1.0 %

Table 2. SSGA final results with different settings

Iter.	Cross. Prob.	Mut. Prob.	Cost Reduction	Improvement Over [1]	Cost for Biconnect
3000	0.85	0.052	19.89 %	9.69 %	0.98 %
3000	0.65	0.020	19.72 %	9.49 %	0.99 %
7000	0.85	0.052	19.90 %	9.69 %	0.99 %

Table 3. SGGA final results with different settings

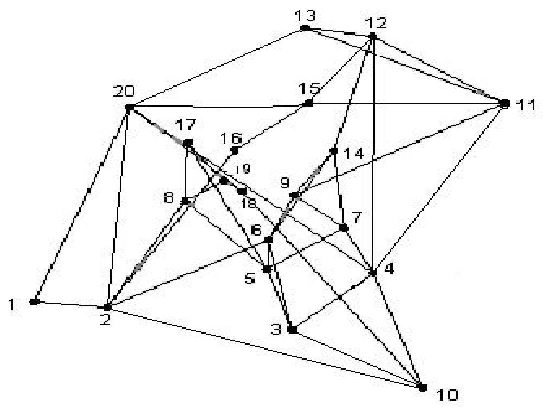


Fig. 11, Initial 20 Nodes Network Topology

To test the performance of the GRGA algorithm under different settings, we tested several operator and parameter value choices. In most cases we concluded that the different options we compared all worked about the same. The final cost of the best solution for different settings is shown in Table 1, 2 and 3. The best results were obtained in conjunction with uniform crossover using 0.85 and 0.052 as the respective values for crossover and mutation probabilities. The computational time spent on the solution was 23 min. and 34 sec. on a SUN Ultra 60 Workstation.

Compared to the initial topology depicted in Fig. 11, the network cost obtained using the GRGA algorithm decreased from an initial value of $46086 to $40080 with no violated constraints after 3000 iterations. This presents an improvement of about 11.5% compared to the initial topology and a 5.98% cost saving compared to [1].

Using the SSGA, after 500 iterations, the algorithm's solution showed a tendency to converge. After 3000 iterations, the network cost is reduced as shown in Fig.12 to $ 38677, this represents an 18.78 % improvement compared to the initial cost, a 3.5% improvement over the GRGA [2] and an 8.79% reduction compared to [1]. This is mainly due to an enforced diversity and small population strategies used, after crossover, members of the population are compared, any duplicate members will be mutated. We found that 38% of chromosomes are identical to another in the population, since duplicate members do not improve the solution, they can be removed. As a result, diversity leading to better solutions is maintained and the algorithm is prevented from premature convergence. Additional advantage is the reduction of the number of fitness tests required; this in turn will reduce the overall solution time of the solution. We found that there was no significant difference between either fitness techniques or selection methods. The different crossover operators and crossover probabilities we tested also all behaved about the same.

The SGGA outperformed both the GRGA and SSGA in convergence behavior and solution quality. An exception is the solution time, while the GRGA and the SSGA spent 28 min. 32sec. and 23 min. 34 sec respectively, the SGGA spent 39 min. 51 sec on the solution. Other than that, the network cost was reduced from an initial $ 47802 to 38290.6 in 3000 iterations. This represent a 19.89% compared to the initial cost, 4.46% compared to the GRGA solution, 1% compared to the SSGA solution and 9.69% compared to [1].

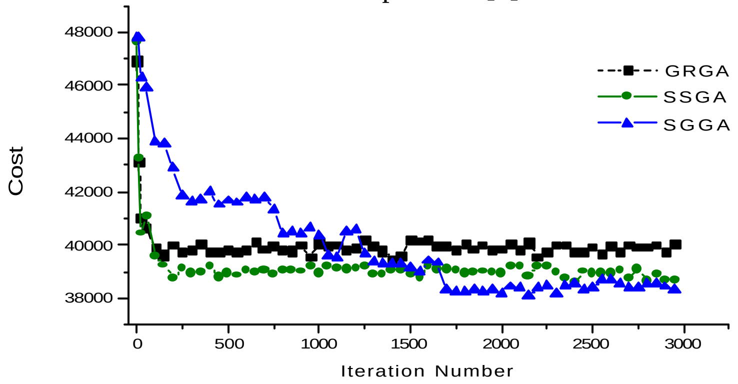


Fig. 12, Cost versus Number of Iteration

The final topologies of the optimal network obtained by the three algorithms are shown in Fig. 13, 14 and 15. It is important to note that due to disconnected topologies incurred after the crossover and mutation operators, link/links are added as explained in Section 3.2.7 to connect back the topologies at an extra cost. In all cases all constraints were respected, including the biconnectivity constraint. We observed an increase in the cost of nearly 0.99 % due to biconnecting the resulted topology after the crossover and mutation operations.

We also tested the three algorithms on a network of 70 nodes. This presented a real challenge to test the efficiency and robustness of the algorithms. While the GRGA failed to find any optimal/near optimal solutions; in fact even finding feasible solution was a very difficult task. The SSGA and SGGA converged to what we believe at this stage is an optimal solution. The problem we faced is the excessive time spent before the termination condition is met. An investigation on improvement strategies on the two algorithms in being carried out and the results will be reported soon.

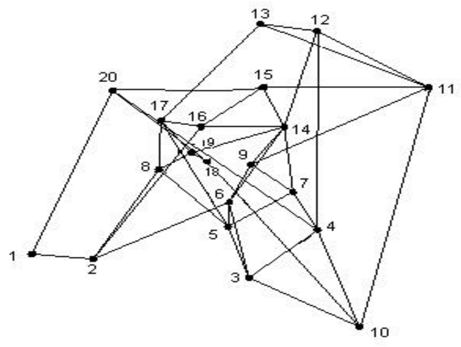


Fig. 13, GRGA Optimal Network Topology

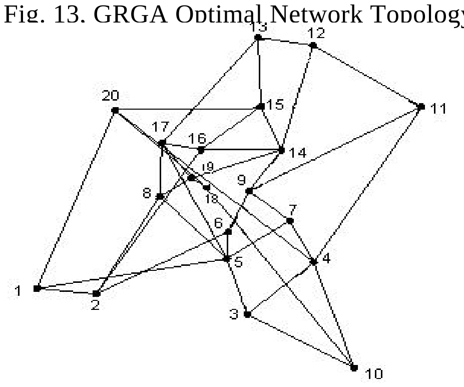


Fig. 14, SSGA Optimal Network Topology

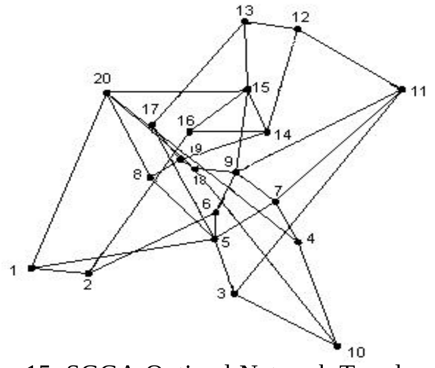


Fig. 15, SGGA Optimal Network Topology

6 Conclusion

The approach described here has been demonstrated to be efficient on a variety of small to medium ATM networks sizes. One early conclusion we reached was that the GRGA, even with elitism, was not very good at finding solutions to the TDCA problem as SSGA or SGGA. In fact, even finding feasible solutions to relatively large problems (networks with more than 70 nodes) proved a difficult challenge. The primary cause of this was premature convergence. The SSGA has proven more successful, particularly at finding feasible solutions. However, the SSGA still had some difficulties finding global optimal solutions. This situation motivated us to develop the SGGA with a large population size to see if there is any possibility to improve the solution further. The possible future work for ATM network planning includes more investigation to improve the capabilities of the SGGA and the expansion of the algorithms to more complicated design issues. The optimal design of virtual path of ATM networks, configuration of digital cross-connect systems to form embedded logical networks from physical facilities networks and survivable network design among others are possible candidate problems to the GA application. Another possibility is hybridizing GAs with robust local search procedures to improve the solution quality and CPU time, and the running of this sequential GA in a parallel fashion, paralleling GAs help in reducing the solution time greatly.

References

[1] Z. Qin, F. F. Wu, N. Law, "Designing B-ISDN Network Topologies Using the Genetic Algorithm", *Proc.* IEEE-MASCOTS'97, 1997, pp. 140-145.

[2] H. Sayoud, K. Takahashi, B. Vaillant, "Designing Communication Networks Topologies Using Steady-State Genetic Algorithms" In Press, *IEEE Comm. Letters*, 2000.

[3] R. Elbaum, M. Sidi, "Topological Design of Local Area Networks Using Genetic Algorithms", IEEE/ACM *Transaction on Networking,* Vol. 4, N^o. 5, pp. 766-778, Oct. 1995

[4] K. Tang, K. Ko, K. Man, S. Kwong, "Topology Design and Bandwidth Allocation of Embedded ATM Networks Using Genetic Algorithms", *IEEE Comm. Letters*, Vol. 2, N^o. 6, June 1998, pp. 171-173

[5] M. Gen, K. Ida, J. R. Kim, " A Spanning Tree Based Genetic Algorithm for Bicreteria Topological Network Design", *Proc. IEEE Intr. Conf. Evol. Compt.*, pp. 15-20, 1998.

[6] D. R. Thompson, G. L. Bilbro, "Comparison of Two Swap Heuristics with a Genetic Algorithm for the Design of ATM Network", *Proc. 7th Inter. Conf. Comp. Comm. & Nets,* pp. 833-837, 1998.

[7] S. Pierre, A. Elguibaoui, "A Tabu-Search Approach for Designing Computer Network Topologies with Unreliable Components", *IEEE Trans. On Reliability*, Vol. 46, N^o. 3, pp. 350-359, Sep. 1997.

[8] B. Denzig, F. Altiparmak, A. Smith, "Efficient Optimization of all Terminal Reliable Networks using an Evolutionary Approach", *IEEE Trans. Reliability*, Vol. 46, pp. 18-26, 1997.

[9] B. Denzig, F. Altiparmak, A. Smith, "Local Search Genetic Algorithm for Optimal Design of Reliable Networks", *IEEE Trans. Evolu. Compt.*, Vol. 1, pp. 179-188, 1997.

[10] D. Deeter, A. Smith,"Heuristic Optimization on Network Design Considering all Terminal Reliability", *Proc. Reliab .& Maintainability. Symp.*, pp. 194-199, 1997.

[11] A. N. Ventetsanopoulos, I. Singh, "Topological Optimization of Communication Networks Subject to Reliability Constraints", *Problem of Contr. Inform. Theory,* Vol. 15, pp. 63-78, 1986.

[12] M. Gerla, J. A. S. Monteiro, and R. Pazos, "Topology Design and Bandwidth Allocation in ATM Nets," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 1253–1262, Oct 1989.

[13] A. Kumar, R. M. Pathak, Y. P. Gupta, "Genetic Algorithm Based Reliability Optimization for Computer Network Optimization", IEEE Trans. Reliability, Vol. 44, pp. 63-72, 1995.

[14] K. F. Man, K. S. Tang, and S. Kwong, "Genetic Algorithms: Concepts and Applications," *IEEE Trans. Ind. Elect.*, Vol. 43, pp. 519–534, Oct. 1996.

[15] L. Berry, B. Murtagh, G. McMachon, L. Welling, "An Integrated GA-LP Approach to Communication Network Design", *Baltzer J.*, pp. 1-16, Jan. 1998

[16] G. A. Walters, D. K. Smith, "Evolutionary Design Algorithm for Optimal Layout of Tree Network", *Engg. Optimization*., Vol. 24, pp. 261-281, 1995.

[17] T. Gruninger, D. Wallace, "Multi-modal Optimization using Genetic Algorithms", *MIT CADlab-Technical Report*: 96.02, 1996.

[18] N. Senin, D. Wallace, N. Borland, "Object-Based Design Modeling and Optimization with Genetic Algorithms", *Proc. GECCO-99*, 1999.

[19] M. B. Wall, "A Genetic Algorithm for Resource Constrained Scheduling", *Ph.D. Thesis,* MIT, 1996.

[20] J. Hopcroft, J. Ullman, "Set Merging Algorithms", *SIAM J. Comput.* Vol. 2, pp. 296-303, 1973.

[21] K. J. Ryul, M. Gen, "Genetic Algorithm for Solving Bicriteria Network Topology Design Problem" *Proc. Evol. Comp., CEC-99*, Vol. 3 , pp. 2272–2279, 1999.

[22] M.C. Sinclair, "Minimum Cost Routing and Wavelength Allocation using a Genetic-Algorithm/Heuristic Hybrid Approach", *Proc. 6th IEE Conf. Telecom.*, pp. 67 –71, 1998.

[23] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd Ed., NY, Springer-Verlag, 1996.

[24] Gen, M, R. Cheng, *Genetic Algorithms & Engineering Design,* John Willey & Sons, NY, 1997.