Journal of Engineering and Applied Sciences 3 (11): 861-864, 2008
ISSN: 1816-949X

Applying a Genetic Algorithm to Telecommunication N/W Design

P. Senthil Kumar and V. Khanaa
Department of Information Technology, University of Bharath, Chennai, India

Abstract: This study presents an Evolutionary Algorithm based approach for the design of telecommunication networks. A permutation-based encoding, uniform Cross-over and problem specific mutation is used for this design optimization problem. We considered a 2-layer network and our aim is to find the Lower Layer (LL) Link capacities for implementing the Upper Layer (UL) Links by minimizing the cost of the Lower Layer. To evaluate the performance of our proposed method, a sample nine-node network is considered and it is compared with other design techniques such as Integer Programming.

Key words: Telecommunication networks, genetic algorithm, evolutionary algorithms, genetic operators, global optimization, encoding, simulated annealing

INTRODUCTION

This study focuses on a specific network design task called network dimensioning in which the network structure is given and the task is to find values of link capacities (i.e., their weights) that are optimal according to a certain criterion. Algorithms are presented for reliable networks (e.g., telecommunication networks like Integrated Services Digital Network (ISDN), Asynchronous Transfer Mode (ATM), Synchronous Digital Hierarchy (SDH)) that are robust to capacity failure. In general, the problem considered here consists in finding sets of flows (in the failure-free state and in each of the considered failure situations) and the cheapest set of link capacities that implement the flows. Such problems are called robust design tasks. The design tasks and methods studied by Pióro (1997).

The use of telecommunication networks to share expensive computer hardware and software resources and also to provide access to Main System from distant locations have been rapidly increasing. Hence, design of telecommunication network has become an essential task in the network industry. While designing a telecommunication network there are several factors such as capacity planning, admission control, wavelength allocation, frequency assignment and others. In our research we consider a specific network design task called network dimensioning in which for a given network structure, the link capacities have to be calculated based on some optimization constraints.

Most contribution to dimensioning of telecommunication networks assume no failure of the network elements. In this study, we consider the dimensioning problems in networks that are robust to failure. The approach can also be generalized to any hierarchical network in which a multi-commodity flow problem is used to model the design. Since, the considered robust design task is an NP-complete, solving it through Integer Programming proves inefficient for networks of larger sizes (Arabas and Kozdrowski, 2001). While employing these techniques, the basic difficulty is introduced by the modularity of link capacities and also due to the nonlinearity nature of the constraints and cost functions, further complexity arises.

Hence, we present the application of an Evolutionary Algorithm which uses Roulette Wheel selection with some problem specific generic operations. The algorithm has been tuned and compares with IP.

MATERIALS AND METHODS

For networks of practical size, the robust design problems are too complex to be solved effectively and hence some kind of decomposition has to be made. Hence, we consider 2 layers, the upper layer that defines the demands between each pair of nodes in this layer and the lower layer, which consists of resources for implementing the demands defined in upper layer.

The multi-commodity flow networks considered in this study are modeled as Multi-graphs. Such a Multi-graph G = (V, E, P) is composed of a set of nodes V, a set of Links E and an incidence relation P defined over the product that specifies how the nodes are connected by the links (Foulds, 1981).

The total cost of the network is defined as a function of link capacities. Y (e), which can be computed as:

Corresponding Author: P. Senthil Kumar, Department of Information Technology, University of Bharath, Chennai, India

$$Z = \sum_{e \in E} \xi(e) y(e)$$

Constraints:

$$\sum_{p \in P(d)} x^{s}_{dp} > n(d)$$

for all d Є D, s Є S and

$$\sum_{d \in D\, p \in P(d)} x^{s}_{dp} < m.\alpha(s,e).y(e)$$

for all, e Є E s Є S.

Network design: In the network design, we distinguish between the UL and the LL network. We assume that the set of nodes of the UL is contained by the set of nodes of the LL. There may exist nodes from LL that are not in UL. LL links are treated as resources that are used to implement the UL network.

Fundamentals of GA: The considered network topologies are generated and optimized by using Genetic Algorithms (GA). GA is a heuristic, adaptive approach for deciding topological problems in net-work planning. GA is developed by John Holland and is based on the principles of the nature selection-surviving of fit individuals and loosing of non fit individuals (Tsenov, 2006).

GAs have been used to solve telecommunication network design problems in a wide variety of contents. However, there is very limited research on using GAs for network dimensioning problem and also examining the GA parameters. This study attempts to address these issues.

Outline of the GA: The method is based on the selection scheme from the $(\mu + \lambda)$ evolution strategy (Fig. 1). The algorithm maintains a population P_t of *chromosomes*, each being a tabulated allocation function. Initially, chromosomes are generated randomly, according to a special procedure that aims to preserve diversity in the population. In the main loop of the algorithm, an offspring population Q_t of λ chromosomes is generated by reproducing randomly selected elements from P_t. Then, chromosomes are randomly mated into pairs, crossed over and mutated. Crossover is performed with a certain probability P_c. A new base population P_{t+1} is generated by selecting μ best chromosomes from the sum of the old base population and the offspring population. The algorithm finds optimal or suboptimal solutions by balancing 2 tendencies-exploration and exploitation. It is believed that increasing population size μ increases (to some extent) explorative properties of GAs.

```
Procedure Genetic Algorithm
Begin
    t:=0
    initialize P0
    evaluate P0
    while (not stop_critertion) do
    begin
        Qt := randomly reproduce Pt
        crossover (x)
        mutate (x)
        Pt+1 := select the best from (Pt U Qt)
        t:= t+1
      end
    end
```

Fig. 1: Outline of the GA

Initial population: The genetic algorithm begins by creating a random initial population. There is very important to obtain the optimal number of individuals in the initial population in order to: give the algorithm enough genetic material for creating "fit" offspring and reduce the working time by finding the optimal solution of any problem. The number of the individuals depends in most cases of the representation of the problem and of the number of the genes in the chromosome (Tsenov, 2006) (Fig. 1).

Number of populations: The number of populations in the algorithm is also important for finding the best solution of a definite problem. This number must be large enough to obtain the optimal solution and at the same time it must be not too large, because of the computing time and the production of too many unused solutions. The number of population depends of the complexity of the problem.

Problem representation: In a network design problem, Encoding is very important to transform the network into the format which can be easily processed by the evolutionary algorithm. Encoding is the process of representing the solution on the search space. Hence, we consider permutation encoding where every chromosome is a string of numbers, which represents nodes in a sequence.

For example, the network structure shown in Fig. 2 and 3 are represented by the chromosome.

In Fig. 2, node 1 is the source and node 8 is the destination. Likewise, in Fig. 3, node 8 is the source and node 9 is the destination.

There are 2 parameters that have to be decided for initialization namely the mutual population size and the procedure to initialize the population. Initially researchers thought that the population size should increase exponentially with the length of the chromosome string in

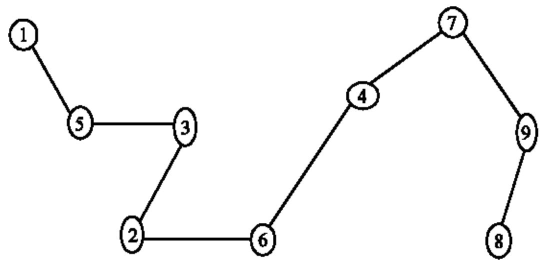

Fig. 2: Chromosome A

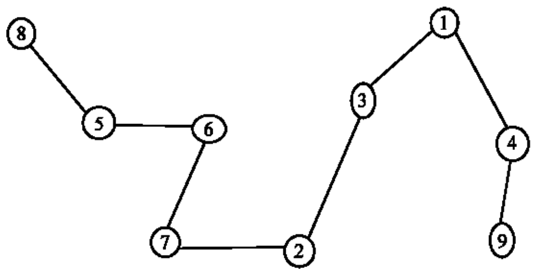

Fig. 3: Chromosome B

order to generate good solutions. Recent studies have shown that satisfactory results can be obtained with a much smaller population size.

There are 2 ways to generate the initial population namely the random initialization and heuristic initialization. In this research random method is employed, In random initialization, we randomly generate an integer from a range of 1 to the number of nodes. The initial chromosomes need not represent a legal or feasible tree.

Fitness function: Genetic algorithm works by maximizing a fitness function, which is formed from the problem objective function and the Constraints. The violation of constraints by the individuals are taken into account through the penalty function. The penalty function penalizes the individual based on the amount of constraints violation. The objective of the penalty function us to lead the optimization algorithm to near-optimal, feasible solution. It is important to allow infeasible solution into the result of breeding between feasible and infeasible solution. There has been a body of work published in Genetic computation on handling constraints.

The extended objective function (1) taking into account the constant link-opening cost implied by the ducts and the penalty function below uses the notion of distance of the solution from feasibility ((x1 - n(d)) + (x2-ma (s,e)y(e)) and a nonlinear penalty (the exponent of 2). With inclusion of penalty function, the fitness function can be written as:

$$F = k/(z + x1\text{-}n(d))2 + (x2\text{-}ma(s,e)y(e))2)$$

GA operations: In each generation, the 3 basic GA operation selection, cross-over and mutation are applied to each individual in the population:

Selection: The selection operator allows individual strings to be copied for possible inclusion in the next generation. There are several selection methods and among those we use roulette, wheel selection to select the individual for the next generation.

Cross-over: Cross-over operator is applied on the selected individuals with probability. Among the various Cross-over operators we consider the Uniform Cross-over Operator.

Mutation: Selection and Cross-over alone can obviously generate a staggering amount of different strings. The GA may find itself converging on strings that are not quite close to the optimum it seeks due to a bad initial population. Some of these problems are overcome by introducing a Mutation Operator into the GA.

Simulated Annealing (SA): The algorithm of the Simulated Annealing (SA) is an approach that integrates most of the local search algorithms. These algorithms accept the next step only when it reduces the cost. So they reach a local minimum and stop searching (Kirkpatrick *et al*., 1983). An essential feature of simulated annealing is that it can climb out from a local minimum, since it can accept worse neighbors at the next step. Such an acceptance happens with a probability that is smaller if the neighbor quality is worse. The probability of the acceptance can be presented as follows:

$$p\{accept\} = \begin{Bmatrix} 1, \text{if } \Delta \le 0 \\ \exp(-\Delta/T), \text{if } \Delta > 0 \end{Bmatrix}$$

where:

Δ = The cost change
Ò = A control parameter that is called temperature

There are 4 problems by the initializing of the algorithm-defining the initial temperature, defining the cooling schedule, defining the number of iterations on each temperature step and stop criterion (Tsenov, 2006).

RESULTS AND DISCUSSION

To evaluate the performance of the proposed GA based approach for Network Design problem, a G-node network is considered. The cost matrix is given in the Table 1.

Table 1: Link cost matrix

Cij	1	2	3	4	5	6	7	8	9
1	0	2	3	4	5	6	7	8	9
2	2	0	11	12	13	14	15	16	17
3	3	11	0	20	21	22	23	24	25
4	4	12	20	0	29	30	31	32	33
5	5	13	21	29	0	38	39	40	41
6	6	14	22	30	38	0	47	48	49
7	7	15	23	30	39	47	0	50	57
8	8	16	24	31	40	48	36	0	65
9	9	17	25	32	41	49	37	65	0

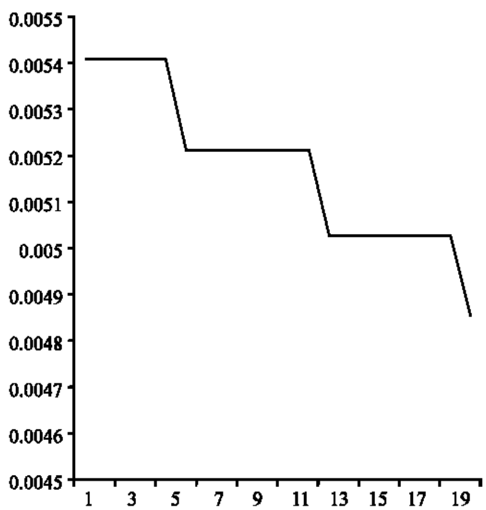

Fig. 4: GA progress

0.005405	0.004854	0.005208	0.005208	0.005208
0.005208	0.005025	0.005025	0.005405	0.005405
0.005405	0.005025	0.005025	0.005208	0.005025
0.005025	0.005025	0.005208	0.005208	0.005405

Ps = 10, Nodes = 9

In each generation, uniform crossover and exchange mutation are applied. While applying GA to thesis problem, the following GA parameters were used.

Maximum generation	20
Population size	10
Crossover probability	0.5
Mutation probability	0.1

The progress of the GA over the 20 iteration is given in Fig. 4. The progress of the SA over the 19 iteration is given in Fig. 5.

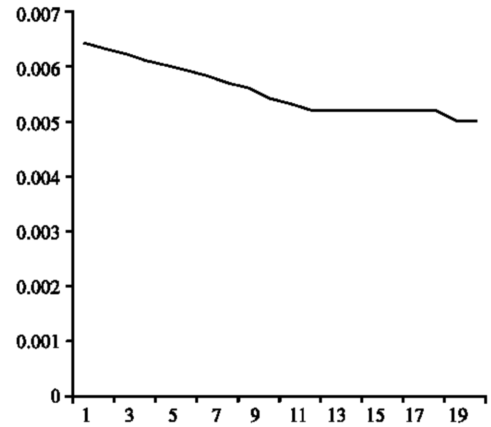

Fig. 5: SA progress

0.006405	0.006305	0.006205	0.006105	0.006005
0.005901	0.005801	0.005701	0.005403	0.005301
0.005201	0.005201	0.005201	0.005201	0.005201
0.005201	0.005201	0.005001	0.005001	

CONCLUSION

For the G-node network, the results obtained by using GA and SA are tabulated. From our observations, the results obtained by GA produces good results when compared to SA for larger networks and for smaller networks SA performs better than GA.

REFERENCES

Arabas, J. and S. Kozdrowski, 2001. Applying an evolutionary algorithm to telecommunication network design. Evolutionary Computation IEEE. Trans., 5 (4): 302-322.

Foulds, L.R., 1981. A multi-commodity flow network design problem. Transportation Research Part B: Methodological, 15 (4): 273-283.

Kirkpatrick, S., C.D. Gelatt, Jr. and M.P. Vecchi, 1983. Optimization by Simulated Annealing, 220: 671-670.

Pióro, M., 1997. Robust design problems in telecommunication networks. In: Proceedings of the 15th International Teletraffic Congress. New York, Elsevier, pp: 1067-1076.

Tsenov, A., 2006. Simulated Annealing and Genetic Algorithm in Telecommunications Network Planning. Int. J. Computat. Intell., 2 (4): 240-245.