Fuzzy Evolutionary Hybrid Metaheuristic for Network Topology Design

Habib Youssef, Sadiq M. Sait, and Salman A. Khan

Department of Computer Engineering, King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia
{youssef,sadiq,salmana}@ccse.kfupm.edu.sa
http://www.ccse.kfupm.edu.sa

Abstract. Topology design of enterprise networks is a hard combinatorial optimization problem. It has numerous constraints, several objectives, and a very noisy solution space. Besides the NP-hard nature of this problem, many of the performance metrics of the network can only be estimated, given their dependence on many of the dynamic aspects of the network, e.g., routing and number and type of traffic sources. Further, many of the desirable features of a network topology can best be expressed in linguistic terms, which is the basis of fuzzy logic. In this paper, we present a fuzzy evolutionary hybrid metaheuristic for network topology design. This approach is **dominance preserving** and scales well with larger problem instances and a larger number of objective criteria. Experimental results are provided.

1 Introduction

A typical enterprise network provides communication services to a large number of hosts, such as mainframe computers, mini systems, workstations, PCs, printers, etc., [1]. Network active elements such as routers, switches, and hubs are used to interconnect these computers and peripherals. The network topology is governed by several constraints. Geographical constraints dictate the breakdown of such internetworks into smaller parts or groups of nodes, where each group makes up what is called a LAN. A LAN consists of all network elements which do not include routers or layer-3 switches. Routers delineate the boundaries of LANs. Communication services of a modern organization are centered around a structured campus network, which consists of a backbone interconnecting a number of LANs via routers or layer-3 switches. Further, the nodes of a LAN may be subdivided into smaller parts, called *LAN segments* (see Fig. 1). Overdimensioning a network is easy; however, designing a cost-optimized network is always very hard. Hardness is a function of the size, the constraints, and obviously the cost parameters to tradeoff. Furthermore, with many cost parameters and constraints, the notion of optimality is not clear. A more reasonable approach is to seek a solution that possesses a set of desirable properties and do not violate some well established design principles. Examples of these principles are:

E. Zitzler et al. (Eds.): EMO 2001, LNCS 1993, pp. 400–415, 2001.

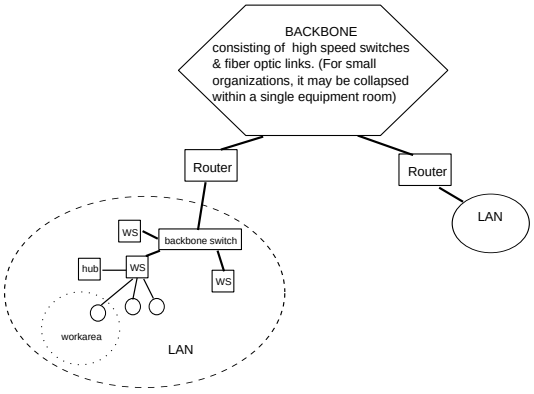

Fig. 1. A typical Campus Network (WS represents workgroup switch).

- There is a physical path between any two nodes.
- The number of hops between any two stations does not exceed a given threshold.
- Only a given small fraction of links have utilization levels below some threshold.

A category of algorithms that were found to be effective for such problems are iterative metaheuristics. These allow you to walk the state space of solutions while evaluating each solution against any desirable set of properties. These meta-heuristics are characterized by hill climbing property that allows occasional acceptance of inferior solutions [2]. Heuristics like genetic algorithm [3], simulated annealing [4], tabu search [5], simulated evolution [6], and stochastic evolution [7] are examples of stochastic iterative heuristics. Detailed description of these heuristics can be found in [2], and an interesting classification of some of them is given in [5].

In this work we propose a hybrid meta-heuristic for the topology design problem which follows the search strategy of Simulated Evolution (SE) algorithm. SE is a memoryless meta-heuristic, where the walk through the state space is heavily influenced by the allocation operator. The memoryless nature of the search usually results in partial revisiting of areas of the state space. To minimize the effect of such undesirable behavior, the allocation step of SE is implemented while following tabu search approach.

2 Background

Many combinatorial optimization problems can be formulated as follows [7]: *Given a finite set M of distinct movable elements and a finite set L of locations, a state is defined as an assignment function $S : M \rightarrow L$ satisfying certain constraints.* The topology design problem fits this generic model. For this problem,

given a set of links $E = \{e_1, e_2, ..., e_n\}$ and a set of locations $L = \{0,\ 1\}$, where $L(e_i) = 1$ iff link e_i belongs to the topology and $L(e_i) = 0$ otherwise. We seek to find an assignment $S : E \rightarrow L$ which corresponds to feasible topology of desirable properties.

Unlike constructive algorithms, which produce a solution only at the end of the design process, iterative algorithms produce numerous solutions during the course of their search. In order to compare alternative topologies, the cost of each topology is estimated for the objectives under consideration. Important objectives are the minimization of monetary cost, network latency, and maximum number of hops between any source-destination pair. Most of the objectives and constraints depend on several aspects such as network flow dynamics, technology trends, strategic commercial goals, etc., that can best be expressed in linguistic terms, which is the basis of fuzzy logic. In this work, the cost function, constraints, as well as some of the SE algorithm operators are implemented using fuzzy algebra [8].

2.1 SE Algorithm

Simulated Evolution (SE) is a stochastic evolutionary search strategy that falls in the general category of meta-heuristics. It was first proposed by Kling and Banerjee in [6]. SE adopts the generic state model described above, where a solution is seen as a population of movable elements.

Starting from a given initial solution, SE repetitively executes the following three steps in sequence: **evaluation**, **selection**, and **allocation**, until certain stopping conditions are met. The pseudo-code of the SE algorithm is given in Fig. 2. The **evaluation** step estimates the **goodness** of each element in its current location. The goodness of an element is a ratio of its optimum cost to its actual cost estimate, and therefore belongs to the interval [0,1]. It is a measure of how near each element is to its optimum position. The higher the goodness of an element, the closer is that element to its optimum location with respect to the current configuration. In **selection** step, the algorithm probabilistically selects elements for relocation. Elements with low goodness values have higher probabilities of getting selected. A selection *bias* (B) is used to compensate for errors made in the estimation of goodness. Its objective is to inflate or deflate the goodness of elements. A high positive value of *bias* decreases the probability of selection and vice versa. Large selection sets also degrade the solution quality due to uncertainties created by large perturbations. Similarly, for high bias values the size of the selection set is small, which degrades the quality of solution due to limitations of the algorithm to escape local minima. A carefully tuned bias value results in good solution quality and reduced execution time [6].

Elements selected during the **selection** step are assigned to new locations in the **allocation** step with the hope of improving their goodness values, and thereby reducing the overall cost of the solution. Allocation is the step that has most impact on the quality of the search performed by the SE algorithm. A completely random allocation makes the SE algorithm behave like a random walk. Therefore, this operator should be carefully engineered to the problem instance

```
Simulated_Evolution(B, Φ_initial, StoppingCondition)
NOTATION
B= Bias Value.
Φ= Complete Solution.
e_i= Individual link in Φ.
O_i= Lower bound on cost of i^th link.
C_i= Current cost of i^th link in Φ.
g_i= Goodness of i^th link in Φ.
S= Queue to store the selected links.
ALLOCATE(e_i, Φ_i)=Function to allocate e_i in partial solution Φ_i
Begin
Repeat

    EVALUATION:   ForEach e_i ∈ Φ DO
                  begin
                      g_i = O_i / C_i
                  end
    SELECTION:    ForEach e_i ∈ Φ DO
                  begin
                      IF Random > Min(g_i + B, 1)
                      THEN
                          begin
                              S = S ∪ e_i; Remove e_i from Φ.
                          end
                  end
          Sort the elements of S
    ALLOCATION:   ForEach e_i ∈ S DO
                  begin
                      ALLOCATE(e_i, Φ_i)
                  end

Until  Stopping Condition is satisfied
Return Best solution.
End (Simulated_Evolution)
```

Fig. 2. Structure of the simulated evolution algorithm.

and must include domain-specific knowledge. Different constructive allocation schemes are proposed in [6].

Though SE falls in the category of meta-heuristics such as simulated annealing (SA) and genetic algorithm (GA), there are significant differences between these heuristics (see [2]). A classification of meta-heuristics proposed by Glover and Laguna [5] is based on three basic features: (1) the use of adaptive memory where the letter A is used if the meta-heuristic employs adaptive memory and the letter M is used if it is memoryless; (2) the kind of neighborhood exploration, where the letter N is used if the meta-heuristic performs a systematic neighborhood search and the letter S is used if stochastic sampling is followed; and (3) the number of current solutions carried from one iteration to the next, where the digit 1 is used if the meta-heuristic maintains a single solution, and the letter P is used if a parallel search is performed with a population of solutions of cardinality P. For example, according to this classification, Genetic algorithm is M/S/P, tabu search is A/N/1, and both simulated annealing and simulated evolution are M/S/1. The heuristic proposed in this work is A/S/1.

2.2 Fuzzy Logic

Fuzzy Logic is a mathematical discipline invented to express human reasoning in rigorous mathematical notation. Unlike classical reasoning in which a proposition is either true or false, fuzzy logic establishes approximate truth value of proposition based on linguistic variables and inference rules. A *linguistic variable* is a variable whose values are words or sentences in natural or artificial language [8]. By using hedges like 'more', 'many', 'few' etc., and connectors like AND, OR, and NOT with linguistic variables, an expert can form *rules*, which will govern the approximate reasoning.

During the topology design process, some desirable objectives, such as the delay, can only be imprecisely estimated. Fuzzy logic provides a rigorous algebra for dealing with imprecise information. Furthermore, it is a convenient method of combining conflicting objectives and expert human knowledge. From the pseudocode of the SE algorithm given in Fig. 2, it is clear that there are two phases of the algorithm which could be modeled to include multiple objectives. These phases are **evaluation** and **allocation**. We have used fuzzy logic based reasoning in these two phases.

3 Assumptions and Notation

In this work, we have assumed the following:

- The (x, y) location of each host is given.
- All hosts have either Ethernet (10 or 100 Mbps) or Token Ring (4 or 16 Mbps) interfaces.
- The traffic rates generated among pairs of hosts are assumed known.
- Vertical cabling (interconnection of local sites to backbone switches) is implemented with fiber optic cables.
- Horizontal cabling portion (cabling within the work area/local site) is implemented with Category 5 UTP (or STP for Token-Ring).
- The root node is a switch acting as a collapsed backbone with given required interfaces.
- There is a user specified limit on the number of network addresses per subnet.
- Maximum allowed utilization of any link should not exceed a desired threshold (e.g. 60 %).

For the following sections, we shall use the notation given below:

n	number of clusters/local sites.
m	number of LAN segments in a cluster.
T	$n \times n$ local site topology matrix where $t_{ij} = 1$, if local sites i and j are connected and $t_{ij} = 0$ otherwise.
λ_i	traffic on link i.
$\lambda_{max,i}$	capacity of link i.
L	number of links of the proposed topology.

D_{nd}	average delay between any source destination pair.
P_i	maximum number of clusters which can be connected to device i.
γ_{ij}	external traffic between clusters i and j.
γ	overall external traffic.

4 Problem Statement

We seek to find a feasible topology of near optimum *overall cost*. A feasible topology is one that satisfies design constraints. Optimality of a topology is measuredwith respect to three objectives: monetary cost, average network delay per packet (network latency), and maximum number of hops between any source-destination pair.

Three important constraints are considered.

1. The first set of constraints is dictated by bandwidth limitation of the links. A good network would be one in which links are "reasonably" utilized, otherwise this would cause delays, congestion, and packet loss. Thus the traffic flow on any link i must never exceed a threshold value:

$$\lambda_i < \lambda_{max,i} \quad i = 1, 2, ..., s \tag{1}$$

   where s is the total number of links present in the topology.
2. The second constraint is that the number of clusters attached to a network device i must not be more than the port capacity P_i of that device.

$$\sum_{j=1}^{n} t_{ij} < P_i \qquad i = 1, 2, ..., n \qquad \forall i \neq j \tag{2}$$

3. The third set of constraints express the designer's desire to enforce certain hierarchies on the network devices. For example, one might not allow a hub to be the parent of a router or backbone device.

Below, we describe the objective criteria used to measure the goodness of a given topology.

Monetary cost: The goal is to find the topology with minimum possible cost, while meeting all the requirements and constraints. The cost of the cable and the cost of the network devices are the two main entities affecting the monetary cost, therefore:

$$cost = (l \times c_{cable}) + (c_{nd}) \tag{3}$$

where l represents the total length of cable, c_{cable} represents the cost per unit of the cable used, and c_{nd} represents the combined costs of all the routers, switches, and hubs used.

Average Network Delay: The second objective is to minimize the average network delay, while considering the constraints and requirements.To devise a suitable function for average network delay, we approximate the behavior of a link and network device by an M/M/1 queue [9].The delay per bit due to the network device between local sites i and j is $B_{i,j} = \mu b_{i,j}$, where $\frac{1}{\mu}$ is the average packet size in bits and $b_{i,j}$ is the delay per packet. If γ_{ij} is the total traffic through the network device between local sites i and j, then the average delay due to all network devices is:

$$D_{nd} = \frac{1}{\gamma} \sum_{i=1}^{d} \sum_{j=1}^{d} \gamma_{ij} B_{ij} \tag{4}$$

where d is the total number of network devices in the network.Thus, the total average network delay is composed of delays of links and network devices and is given by [9]

$$D = \frac{1}{\gamma} \sum_{i=1}^{L} \frac{\lambda_i}{\lambda_{max,i} - \lambda_i} + \frac{1}{\gamma} \sum_{i=1}^{d} \sum_{j=1}^{d} \gamma_{ij} B_{ij} \tag{5}$$

Maximum number of hops between any source-destination pair: The maximum number of hops between any source-destination pair is also another objective to be optimized. A hop is counted as the packet crosses a network device.

5 Proposed Algorithm and Implementation Details

This section describes our proposals of fuzzification of different stages of the SE algorithm. We confine ourselves to tree design. Trees are minimal and provide unique path between every pair of local sites. Further, the design of a general mesh topology usually starts from a near optimal constrained spanning tree.

5.1 Initialization

The initial spanning tree topology is generated randomly, while keeping into account the feasibility constraints mentioned earlier.

5.2 Proposed Fuzzy Evaluation Scheme

The **goodness** of each individual is computed as follows. In our case, an individual is a **link** which interconnects the devices of two local sites (at the backbone level) or two network devices (at the local site level). In the *fuzzy evaluation scheme*, monetary cost and optimum depth of a link (with respect to the root) are considered fuzzy variables. Then the goodness of a link is characterized by the following rule.

Rule 1: IF a link is *near optimum cost* AND *near optimum depth*
THEN it has *high goodness*.

Here, *near optimum cost*, *near optimum depth*, and *high goodness* are linguistic values for the fuzzy variables cost, depth, and goodness. Using and-like compensatory operator [10], Rule 1 translates to the following equation for the fuzzy goodness measure of a link l_i.

$$g_{l_i} = \mu^e(l_i) = \alpha^e \times min(\mu_1^e(l_i), \mu_2^e(l_i)) + (1 - \alpha^e) \times \frac{1}{2} \sum_{i=1}^{2} \mu_i^e(l_i) \qquad (6)$$

The superscript e stands for **evaluation** and is used to distinguish similar notation in other fuzzy rules. In (6), $\mu^e(l_i)$ is the membership in the fuzzy set of *high goodness links* and α^e is a constant. The $\mu_1^e(l_i)$ and $\mu_2^e(l_i)$ represent memberships in the fuzzy sets *near optimum monetary cost* and *near optimum depth*.

In order to find the membership of a link with respect to *near optimum monetary cost*, we proceed in following manner. From the cost matrix, which gives the costs of each possible link, we find the minimum and maximum costs among all the link costs. We take these minimum and maximum costs as the lower and upper bounds and call them "LCostMin" and "LCostMax" respectively and then find the membership of a link with respect to these bounds. Furthermore, in this work, we have normalized the monetary cost with respect to "LCostMax". The required membership function is represented as depicted in Fig. 3, where $x - axis$ represents $\frac{LCost}{LCostMax}$, $y - axis$ represents the membership value, $A = \frac{LCostMin}{LCostMax}$, and $B = \frac{LCostMax}{LCostMax} = 1$. This normalization enables us to use the same membership function for all topology design instances.

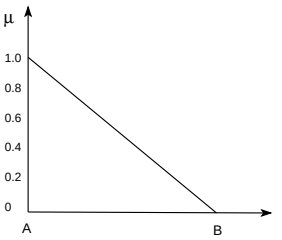

Fig. 3. Membership function for the objective to be optimized.

In the same manner, we can find the membership of a link with respect to *near optimum depth*. The lower limit, which we call "LDepthMin" is taken to be a depth of 1 with respect to the root. The upper bound, which we call "LDepthMax" is taken to be 1.5 times of the maximum depth generated in the

initial solution or a maximum of a user specified limit.[1] For example, if in the initial solution, the maximum depth turns out to be 4, then "LDepthMax" for the depth membership function would be 6. This is done to give chance to links which may have more depth than the one in the initial solution. If we take the initial solution maximum depth as "LDepthMax", then in the following iterations some links with higher depths will have a membership value of zero (with respect to depth membership function) and thus they will not be able to play any role as far as depth is concerned. However, due to technological limitations, we have limited the maximum possible depth to 7, in the case when "LDepthMax" turns out to be more than 4. The reason for having the maximum depth of 7 is that the hop limit for RIP is 15. This means that if a maximum depth of 7 is taken, then in the worst case we would have a total of 14 hops from a source to a destination. The membership function with respect to *near optimum depth* can be represented as illustrated in Fig. 3, where $x-axis$ represents $LDepth$, $y-axis$ represents the membership value, $A = LDepthMin$, and $B = LDepthMax$.

5.3 Selection

In this stage of the algorithm, for each link l_i in current tree topology, where i = 1,2,..., n-1, a random number $RANDOM \in [0, 1]$ is generated and compared with $g_i + B$, where B is the selection bias. If $RANDOM > g_i + B$, then link l_i is selected for allocation and considered removed from the topology. Bias B is used to control the size of the set of links selected for removal. A bias methodology called *variable bias* [11] has been used in this paper. The *variable bias* is a function of *quality of current solution*. When the overall solution quality is poor, a high value of bias is used, otherwise a low value is used. Average link goodness g_i is a measure of how many "good" links are present in the topology. The bias value changes from iteration to iteration depending on the quality of solution. The *variable bias* is calculated as follows:

$$B_k = 1 - G_k$$

where B_k is the bias for k^{th} iteration and G_k is average goodness of all the links at the beginning of iteration k.

5.4 Proposed Fuzzy Allocation Scheme

During the **allocation** stage of the algorithm, the selected links are removed from the topology one at a time. For each removed link, new links are tried in such a way that they result in overall better solution. Before the allocation step starts, the selected links are sorted according to their goodness values in ascending order.

[1] This user specified limit may be a design constraint, e.g., if each hop represents a router that uses Routing Information Protocol (RIP) then a limit would be 7, i.e., a branch of the tree should not have more than 7 routers.

In the *fuzzy allocation scheme*, the three criteria to be optimized are combined using fuzzy logic to characterize a good topology. The reason for using fuzzy logic is that the characterization of a good topology with respect to several criteria is usually based on heuristic knowledge which is acquired through experience. Such knowledge is most conveniently expressed in linguistic terms, which constitute the basis of fuzzy logic. For the problem addressed in this paper, a good topology is one that is characterized by a low monetary cost, low average network delay, and a small maximum number of hops. In fuzzy logic, this can easily be stated by the following fuzzy rule:

> Rule 2: **IF** a solution X has *low monetary cost* AND *low average network delay* AND *low maximum number of hops between any source-destination pair* **THEN** it is a *good topology.*

The words "low monetary cost", "low average network delay", "low maximum number of hops", and "good topology" are linguistic values, each defining a fuzzy subset of solutions. For example, "low average network delay" is the fuzzy subset of topologies of low average network delays. Each fuzzy subset is defined by a membership function μ. The membership function returns a value in the interval [0,1] which describes the degree of satisfaction with the particular objective criterion. Using the and-like ordered weighted averaging operator [10], the above fuzzy rule reduces to the following equation.

$$\mu^a(x) = \beta^a \times min(\mu_1^a(x), \mu_2^a(x), \mu_3^a(x)) + (1 - \beta^a) \times \frac{1}{3} \sum_{i=1}^{3} \mu_i^a(x) \qquad (7)$$

where $\mu^a(x)$ is the membership value for solution x in the fuzzy set *good topology* and β^a is a constant in the range [0,1]. The superscript a stands for allocation. Here, μ_i^a for $i = \{1,2,3\}$ represents the membership values of solution x in the fuzzy sets *low monetary cost*, *low average network delay*, and *low maximum number of hops between any source-destination pair* respectively. The solution which results in the maximum value for (7) is reported as the best solution found by the SE algorithm.

Below we describe how to get the membership functions for the three criteria mentioned above.

Membership Function for Monetary Cost. First, we determine two extreme values for monetary cost, i.e., the minimum and maximum values. The minimum value, "TCostMin", is found by using the Esau-Williams algorithm [12], with all the constraints completely relaxed. This will surely give us the minimum possible monetary cost of the topology. The maximum value of monetary cost,"TCostMax", is taken to be the monetary cost generated in the initial solution. The monetary cost is normalized with respect to "TCostMax". The corresponding membership function is shown in Fig. 3, where $x - axis$ represents $\frac{TCost}{TCostMax}$, $y - axis$ represents the membership value, $A = \frac{TCostMin}{TCostMax}$, and $B = \frac{TCostMax}{TCostMax} = 1$.

Membership Function For Average Network Delay. We determine two extreme values for average network delay. The minimum value, "TDelayMin", is found by connecting all the nodes to the root directly, ignoring all the constraints and then calculating the average network delay using (5). The maximum value of average delay,"TDelayMax", is taken to be the average delay generated in the initial solution. The average delay is normalized with respect to "TDelayMax". The membership function is shown in Fig. 3, where $x - axis$ represents $\frac{TDelay}{TDelayMax}$, $y - axis$ represents the membership value, $A = \frac{TDelayMin}{TDelayMax}$, and $B = \frac{TDelayMax}{TDelayMax} = 1$.

Membership Function For Maximum Number of Hops. Again, two extreme values are determined. The minimum value, "THopsMin", is taken to be 1 hop, which will be the minimum possible in any tree. The maximum value,"THopsMax", is taken to be the maximum number of hops between any source-destination pair generated in the initial solution. The membership function is shown in Fig. 3, where $x - axis$ represents $THops$, $y - axis$ represents the membership value, $A = THopsMin$, and $B = THopsMax$.

In the proposed allocation scheme, all the selected links are removed one at a time and trial links are placed for each removed link. We start with the head-of-line link, i.e. the link with the worst goodness. We remove this link from the topology. This divides the topology into two disjoint trees. Now the placing of trial links begins. In this work, the approach to place trial links is as follows. At most ten trial moves (i.e., trial links) are evaluated for each removed link. One point to mention is that for the ten moves, some moves may be invalid. However, we search for only four "valid" moves. Whenever we find four valid moves, we stop, otherwise we continue until a total of ten moves are evaluated (whether valid or invalid). The removal of a link involves two nodes P and Q, of which node P belongs to the subtree which contains the root node and node Q belongs to the other subtree. For the ten moves we make, five of them are greedy and five are random. For the greedy moves, we start with node Q and five *nearest* nodes in the other subtree are tried. For the random moves, we select any two nodes in the two subtrees and connect them. If all the ten moves are invalid, in which case the original link is placed back in its position. The valid moves are evaluated based on (7) and the best move among the ten moves is made permanent. This procedure is repeated for all the links that are present in the set of selected links.

We have implemented two variations of allocation schemes. The first one is the same as has been described above, which we call SE. In the second variation, Tabu Search characteristics have been introduced, details of which follow.

5.5 Tabu Search Based Allocation

Tabu Search (TS) is a general iterative heuristic that is used for solving combinatorial optimization problems. The algorithm was first presented by F. Glover [5].

Table 1. Characteristics of test cases used in our experiments. LCostMin, LCostMax, and TCostMin are in dollars. TDelayMin is in milliseconds. Traffic is in Mbps.

Name	# of Local Sites	LCostMin	LCostMax	TCostMin	TDelayMin	Traffic
n15	15	1100	9400	325400	2.14296	24.63
n25	25	530	8655	469790	2.15059	74.12
n33	33	600	10925	624180	2.15444	117.81
n40	40	600	11560	754445	2.08757	144.76
n50	50	600	13840	928105	2.08965	164.12

A key feature of TS is that it imposes restrictions on the search process, preventing it from moving in certain directions to drive the process through regions desired for investigation [5]. It searches for the best move in the neighborhood of the current solution.

In this work, we have modified the SE algorithm by introducing Tabu Search characteristics in the allocation phase. Recall that in the allocation phase, certain number of moves are made for each link in the selection set and the best move is accepted, making the move (i.e., link) permanent. This newly accepted link is saved in a *tabu list*. Thus our *attribute* is the link itself. The *aspiration criterion* adopted is that if the link that had been made tabu produces a higher membership value than the current one in the membership function "good topology", then we will override the tabu status of the link and make it permanent. This strategy prevents the selection and allocation operators from repetitively removing the same link and replacing it with a link of equal or worse goodness.

5.6 Stopping Criterion

In our experiments, we have used a fixed number of iterations as a stopping criterion. We experimented with different values of iterations and found that for all the test cases, the SE algorithm converges within 4000 iterations or less.

6 Results and Discussion

The SE algorithm described in this paper has been tested on several randomly generated networks. For each test case, the traffic generated by a typical local site was collected from real sites. Other characteristics, such as the number of ports on a network device, its type, etc. were assumed. However, the costs of the network devices and links were collected from vendors. The characteristics of test cases are listed in Table 1. The smallest test network has 15 local sites and the largest has 50 local sites. The hierarchies in which the devices are connected are that backbone switch is at the top, followed by routers, then workgroup switches, and then hubs.

Table 2. Best solution for different tabu list sizes. Monetary cost is in dollars, delay is in milli seconds per packet, and execution time is in minutes.

Test Case	Tabu list size	Monetary Cost	Avg. Delay	Max. Hops
n15	1	298200	2.935	4
	2	297100	2.78	4
	3	294350	3.448	6
	4	298100	3.037	5
	5	296900	3.278	6
n25	3	481745	4.219	8
	4	478690	4.189	9
	5	483210	3.537	6
	6	479915	4.275	9
	7	488400	4.608	9
n33	3	655715	5.772	11
	5	652785	4.77	8
	6	682465	4.19	6
	7	652310	5.95	10
	9	667100	5.087	7
n40	5	785795	4.746	10
	6	798695	8.019	12
	7	783970	4.441	9
	8	786950	5.478	9
	9	790645	5.136	8
n50	4	958995	6.739	14
	5	967110	9.279	14
	7	983020	5.245	11
	8	1075450	5.725	9
	9	971965	7.13	12

6.1 Effect of Tabu Search Based Allocation and Tabu List Size

Table 2 shows the results obtained for the test cases using different tabu list sizes. In this table, monetary cost, average delay, and maximum hops of best solutions are reported along with the respective tabu list size. In the table we notice that as the test case size increases, the tabu list that gives the best solution also increases. For example, in *n15*, tabu list size of 2 gives the best solution. Similarly, best solutions are achieved by tabu list sizes of 5, 6, 7, and 7 in *n25*, *n33*, *n40*, and *n50* respectively.

Table 3 gives the results for different test cases considering the frequency of tabu moves, and the respective tabu list size that gave the best solutions with their execution times. By frequency of tabu moves we mean the number of times a link was found tabu. We record this through a counter called *tabu counter*. The tabu counter only includes the number of tabu links which could not pass the aspiration criteria. It does not count the frequency of links which were actually tabu but managed to pass the aspiration criteria. From this figure, it can be seen that the percentage of tabu moves varies between 1% and 10%.

6.2 Comparison of SE and SE_TS

In this section, the results of SE_TS with SE are compared. Table 4 shows the results for SE and best tabu list size SE_TS. The percentage gain shows the

Table 3. Results for best tabu list size. Execution time is in minutes.

Test case	Tabu list size for best solution	Total moves	Tabu moves	% Tabu moves	Exec. time
n15	2	1241	45	3.62	2.25
n25	5	2496	39	1.56	4
n33	6	1352	93	6.878	8
n40	7	4223	233	5.51	26
n50	7	3995	328	8.21	65

improvement achieved by SE_TS when compared to SE. From this table, it is seen that SE_TS performs better than SE as far as monetary cost objective is concerned. In all the test cases, a gain is achieved by SE_TS. For example, a gain of 5.67 % is achieved in case of *n50*. A similar behavior is seen for average network delay metric, where SE_TS achieves gain in all the cases. Similarly, for maximum number of hops metric, a gain is achieved for all the cases except *n50*. However, the loss in maximum hops for *n50* is compensated by the improvement in the monetary cost and delay metrics. As far as the execution time is concerned, it is also comparable.

To compare the quality of search space between SE and SE_TS , we plot different optimization parameters versus iteration count of the algorithms for the test case *n40* (best tabu list size=7 in SE_TS). Fig. 4(a) compares the current monetary cost. From this plot it is clear that SE_TS converges a little faster towards a better solution. Somewhat similar behavior is seen with respect to average network delay and maximum hops parameters in Figs. 4(b) and (c), where we see that SE_TS performs better than SE. The reason SE_TS has better performance than SE is the following. In SE, since the search space for valid solutions is limited, it happens that after some iterations, same moves are repeated and thus the algorithm keeps searching in the same search space most of the time, while in SE_TS, more search space is covered because previous moves remain tabu for some time, causing the algorithm to diversify the search into another subarea. Recall that in the allocation phase, four valid moves are evaluated for each link in selection set and the best move (link) is made permanent. This new link is also saved in the tabu list simultaneously. However, it may happen that this new link may become "bad" (in terms of evaluation function) in the following iterations, upon which it is removed. But it may be possible that this link may become good again after one or more iterations, but since it is in the tabu list, it will not be chosen to be placed again, thus giving room to other links to be chosen. In general we see that SE_TS achieves better quality solution than SE.

7 Conclusion

In this paper we have presented a novel approach for topology design of campus networks based on fuzzy simulated evolution algorithm with two variations in

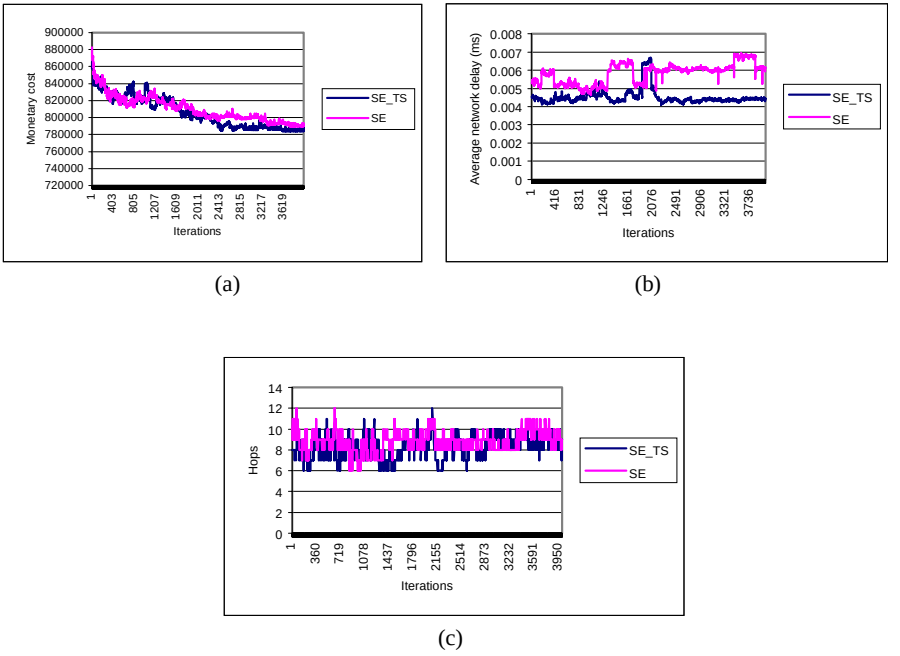

Fig. 4. Comparison of SE and SE_TS for n40.

Table 4. Comparison of SE and SE_TS. C = Cost in dollars, D = Delay in milli seconds per packet, H = hops, T = execution time in minutes, TL= Tabu list size. Percentage gain shows improvement achieved by SE_TS compared to SE.

Case	SE					SE_TS				% Gain		
	C	D	H	T	TL	C	D	H	T	C	D	H
n15	305500	4.135	7	1	2	297100	2.78	4	2.25	2.7	32.8	42.8
n25	512415	4.37	7	4.4	5	483210	3.537	6	4	5.7	19.1	14.28
n33	702815	5.319	7	17	6	682465	4.19	6	8	2.89	21.2	14.28
n40	789625	5.529	9	42	7	783970	4.441	9	26	0.72	19.7	0
n50	1042080	8.236	10	62	7	983020	5.245	11	65	5.67	36.3	-9.1

the allocation scheme. Results obtained for the test cases considered suggest that fuzzy simulated evolution algorithm with tabu search allocation is a robust approach to this problem, and was always able to find good quality feasible solutions.

Acknowledgments. Authors acknowledge King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, for all support.

References

1. Habib Youssef, Sadiq M. Sait, and Osama A. Issa. Computer-Aided Design of Structured Backbones. In *15th National Computer Conference and Exhibition*, pages 1–18, October 1997.
2. Sadiq M. Sait and Habib Youssef. *Iterative Computer Algorithms and their Application to Engineering.* IEEE Computer Society Press, 1999.
3. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Company, INC., 1989.
4. S. Kirkpatrick, C. Gelatt Jr., and M. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):498–516, May 1983.
5. Fred Glover and Manuel Laguna. *Tabu Search.* Kluwer Academic Publishers, 1997.
6. Ralph M. Kling and Prithviraj Banerjee. ESP: Placement by Simulated Evolution. *IEEE Transactions on Computer-Aided Design*, 8(3):245–255, March 1989.
7. Y. Saab and V. Rao. Stochastic Evolution: A Fast Effective Heuristic for some Generic Layout Problems. In *27th ACM/IEEE Design Automation Conference*, pages 26–31, 1990.
8. L. A. Zadeh. Fuzzy Sets. *Information Contr.*, 8:338–353, 1965.
9. R. Elbaum and M. Sidi. Topological Design of Local-Area Networks Using Genetic Algorithm. *IEEE/ACM Transactions on Networking*, pages 766–778, October 1996.
10. Ronald Y. Yager. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking. *IEEE Transactions on Systems, Man, and Cybernatics*, 18(1):183–190, Jan 1988.
11. Ali S. Hussain. *Fuzzy Simulated Evolution Algorithm for VLSI Cell Placement.* MS Thesis, King Fahd University of Petroleum and Minerals, 1998.
12. L. R. Esau and K. C Williams. On teleprocessing system design. A method for approximating the optimal network. *IBM System Journal*, 5:142–147, 1966.