A FUZZY EVOLUTIONARY ALGORITHM FOR TOPOLOGY DESIGN OF CAMPUS NETWORKS

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

Department of Computer Engineering
King Fahd University of Petroleum and Minerals
Saudi Arabia

الخلاصــة

إن التصميم الطبولوجي لشبكات المجمعات يتطلب حلولا تقريبية ضمن إطار من المتطلبات الصعبة، حيث يشتمل التصميم على تحديد عدد العناصر الفعالة ونوعها وموقعها وخطوط وصلها ضمن الحدود الفيزيائية والتقنية، مما يوجب إيجاد نقاط تقارب بين العديد من المتطلبات المتعارضة، وأهم هذه المتطلبات التكلفة المالية، وزمن النقل في الشبكة، وعدد القفزات بين نقاط الاتصال، وموثوقية الشبكة إضافة إلى ذلك فإن هذه المتطلبات غير محددة بشكل دقيق بسبب صعوبة التنبؤ بحركة البيانات داخل الشبكة. وفي مثل هذه الحالات يوفر المنطق المبهم إطاراً رياضياً مناسباً لحل المشكلة.

وسوف نقدم في هذا البحث خوارزمية للتصميم الطبولوجي لشبكات المجمعات يعتمد على خوارزم التطور المحاكي. ولتفعيل عملية البحث في مرحلة التوزيع قمنا باستخدام خواص للبحث مستقاة من خوارزم Tabu Search. وقد قمنا بمقارنة الخوارزم المقترح مع كلتا الخوارزميتين Simulated Annealing و Esau–Williams. وقد أظهرت النتائج في كل الحالات التي تم اختبارها تفوّق خوارزمية التطور المحاكي على الخوارزميتين الأُخريين.

*Address for correspondence:
KFUPM Box 5065
King Fahd University of Petroleum & Minerals
Dhahran-31261, Saudi Arabia
e-mail: {youssef,sadiq,salmana}@ccse.kfupm.edu.sa

ABSTRACT

The topology design of campus networks is a hard constrained combinatorial optimization problem. It consists of deciding the number, type, and location of the active network elements (nodes), and the links. This choice is dictated by physical and technological constraints and must optimize several objectives. Important objectives are monetary cost, network delay, hop count between communicating pairs, and reliability. Furthermore, due to the nondeterministic nature of network traffic and other design parameters, the objective criteria are imprecise. Fuzzy Logic provides a suitable mathematical framework in such a situation. In this paper, we present a Simulated Evolution algorithm for the design of campus network topology. To intensify the search, we have also incorporated Tabu Search-based characteristics in the allocation phase of the SE algorithm. The proposed fuzzy SE algorithm is compared with the Simulated Annealing heuristic. Comparison is also made with Esau–Williams (EW) algorithm, a well known constructive algorithm for the category of problems addressed in this work. Results show that on all test cases, the Simulated Evolution algorithm exhibits a more intelligent search of the solution subspace and was able to find better solutions than Simulated Annealing and Esau–Williams algorithm.

Keywords: Campus Networks, Combinatorial Optimization, Fuzzy Logic, Iterative Heuristics, Network Topology, Simulated Annealing, Simulated Evolution, Tabu Search.

A FUZZY EVOLUTIONARY ALGORITHM FOR TOPOLOGY DESIGN OF CAMPUS NETWORKS

1. INTRODUCTION

A typical campus network consists of an interconnected collection of a relatively large number of nodes. The network nodes fall into two general categories: *end-user nodes* which represent network access points consisting of workstations, personal computers, printers, mainframe computers, *etc.*, and the *network active elements* consisting of various devices such as multiplexers, hubs, switches, routers, and gateways. The active elements and links provide the needed physical communication paths between every pair of end-user nodes.

A good network topology is governed by several constraints. Geographical constraints dictate the break-down of such internetwork into smaller parts or groups of nodes, where each group makes up what is called a LAN. A LAN consists of all the elements that create a networked system up to a router. A campus network is usually made up of a collection of interconnected LANs. Further, the nodes of a LAN may be subdivided into smaller parts, called *LAN segments*, to satisfy other constraints and objectives, for example, minimization of delay, containment of broadcast traffic, and minimization of cabling and equipment cost [1]. The topology design of the LAN itself consists of two main issues: *segmentation*, where LAN segments are defined, and *design of actual topology*, which consists of interconnecting the individual segments. Topology design at LAN level usually consists of interconnecting the LAN segments *via* bridges and layer 2 switches [2, 3].

It is recommended to structure the campus network into three layers (see Figure 1):

(1) Local Access Layer, which provides workgroup access to the network.

(2) Distribution Layer, which provides policy-based connectivity among the workgroups. This layer is implemented with layer 3 switches, routers, and gateways. This is where packet manipulation takes place.

(3) Backbone Layer, which provides high speed optimal transport of data among local sites (the LANs).

Following the above three-layer hierarchy, the design of such a structured campus network can be approached in four steps:

(1) Assignment of end-user nodes/stations to LAN segments.

(2) Assignment of LAN segments to local sites that will make up a single LAN.

(3) Design of the internal structure of each local site (*i.e.*, in what topology the LAN segments of a local site are connected). This step serves also to select appropriate switching equipment.

(4) Backbone design, where the local sites are connected to the backbone. This step also will dictate the required backbone equipment.

1.1. Literature Review and Related Work

Topological design of campus networks is a hard problem [1]. Even the design of a LAN is itself an NP-hard problem [2–4]. The state space is of exponential complexity. For example, for a network with n nodes, there can be as many as $2^{\frac{n(n-1)}{2}}$ different topologies. Even for $n = 20$ this number evaluates to more than 10^{56}. Therefore, we have to use approximation methods known as 'heuristics' to focus the search on feasible topologies of desirable features in order to produce good feasible solutions in a reasonable amount of time.

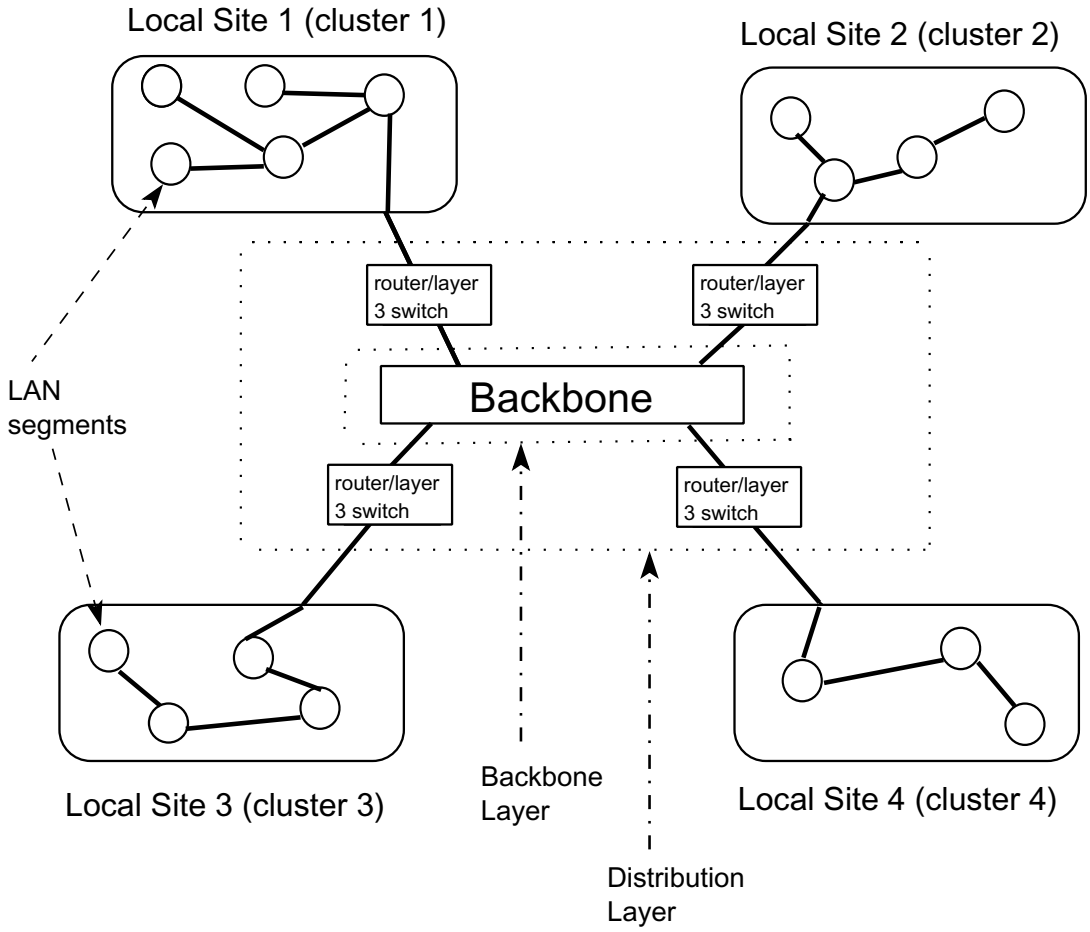

Figure 1. A typical campus network.

There are two categories of heuristics: *constructive* and *iterative.* Constructive heuristics produce a complete solution by making deterministic moves. Constructive techniques are fast but produce in most cases solutions of poor quality. For highly constrained problems, they may even fail to find a feasible solution. Some of the well known constructive algorithms for the constrained minimum spanning tree problem are the Kruskal algorithm [5], the Prim algorithm [6], and the Esau–Williams algorithm [7].

Iterative heuristics attempt to improve a complete solution by making a controlled walk through the state space [8]. Generally we can classify iterative schemes into two subclasses: schemes which only accept good moves, *i.e.*, perturbations leading to better quality solutions, like local search; and schemes which can accept bad moves, like simulated annealing [9], genetic algorithms [10], tabu search [8], and simulated evolution [8]. Such search schemes are said to possess hill-climbing property, enabling them to escape local optima and hence discover better quality solutions.

The use of iterative heuristics for topological network design has been reported in many research papers. The use of Genetic Algorithms (GA) for topological network design has been proposed in [2, 3, 11, 12]. Pierre *et al.* [12] used a genetic algorithm to solve the topological design problem of distributed packet switched networks. The goal was to find a topology that minimizes the communication costs by taking into account constraints such as delay and reliability. Elbaum *et al.* [2] have used GA for designing LANs with the objective of minimizing the average network delay. The constraint they considered is that the flow on any link does not exceed the capacity

of that link. The topology design issues they addressed consist of determination of the number of segments in the network, allocating the users to different segments, and determining the interconnections and routing among the segments. Also, lower bounds on the average network delay have been developed. Gen *et al.* [3] have used GA for topological network design with two criteria which are the network delay and cost based on the weights of links. They have used two main entities which they call service centers (*e.g.*, bridges) and nodes (*i.e.*, users) which are connected to service centers. The constraints they have considered are that the number of nodes connected to a service center must not exceed the capacity of the center and that the traffic flowing through a service center must not exceed its traffic capacity. Dengiz *et al.* [11, 13] focused on large backbone communication network design using genetic algorithm to optimize a network topology. They used the cost and reliability as optimization measures. They called their algorithm GA with knowledge-based steps (GAKBS). Similarly, use of simulated annealing has been reported in [4] where Ersoy *et al.* used it for topological design of interconnected LAN/MAN. The main objective was to minimize the average network delay. They have considered transparent bridges, which are required to form a spanning tree topology. Fetterolf [14] used simulated annealing to design LAN–WAN computer networks with transparent bridges. The simulated annealing algorithm that was proposed generates sequences of neighboring spanning trees and evaluates design constraints based on maximum flow, bridge capacity, and end-to-end delay. In [15], Pierre *et al.* used a Tabu Search algorithm for designing computer network topologies with unreliable components. Their simulation results showed the efficiency and robustness of their algorithm for designing backbone networks interconnecting between 12 and 30 nodes. However, in all the aforementioned studies, realistic assumptions, such as the limitation of number of ports on a networking device, or the hierarchy of network devices, were not considered.

In this work, a simulated evolution (SE) based heuristic [8] is used for topology design of structured campus networks. The proposed heuristic is engineered to seek feasible tree topologies that are minimized with respect to monetary cost, maximum number of hops between any source–destination pair, and average network delay per packet. For assignment of segments to local sites, Augmenting Path algorithm is used [16].

For campus networks, the number of nodes is relatively small and the links are highly reliable, which justifies the use of a tree topology. Further, according to recommended structured cabling standards, the network topology is constrained to be a hierarchical star, *i.e.*, a tree. Hence we target to find a constrained tree topology of desirable quality with respect to the three design objectives. We resort to fuzzy logic to formulate the various objectives in the form of fuzzy rules that will guide the search toward solutions of desirable quality.

2. ASSUMPTIONS AND NOTATION

The term "node" is used to refer to different objects at different levels. At the segment level, a node refers to a network access point (end user equipment), while at the LAN level, a node refers to a segment (shared or switched hub). At the backbone level, a node refers to a LAN, or a 'cluster' (actually the node represents the switch, router, or gateway connecting the LAN to another backbone node).

In this work we assume the following.

- The backbone is assumed to consist of a single node, *i.e.*, we assume a collapsed backbone equipped with required interfaces. The backbone is the *root* of the tree.
- A LAN/cluster is constrained to consist of segments of the same technology such as Ethernet.
- The "capacity" of a network device is equal to the number of interfaces it has.
- The location of a node within a cluster can be represented by its Euclidean (x, y) coordinates with respect to some reference point.
- Each end-user node has a LAN network interface card of a particular technology such as 10/100baseT Ethernet or Token-Ring type [17].

- Between two local sites, only fiber optic cable [18] is used.
- There is a user-specified limit on the number of network addresses per LAN.
- The number of segments is known *a priori*, *i.e.*, users have already been assigned to segments.
- Hubs, switches, routers, and other networking devices cannot be placed in any location. They have designated locations (equipment racks/closets in particular rooms).
- Maximum allowed utilization of any link should not exceed a desired threshold (*e.g.*, 60%).

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

n	number of clusters.
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 in bits per second (bps) on link i.
$\lambda_{max,i}$	capacity in bps of link i.
L	number of links of the proposed topology.
D_{nd}	delay due to network devices.
b_{ij}	the delay per packet.
μ	reciprocal of average packet size in bits.
B_{ij}	the delay per bit due to the network device feeding the link connecting local sites i and j, equal to $\mu b_{i,j}$.
n_i	maximum number of nodes that can be connected to cluster i.
γ_{ij}	external traffic in bps between clusters i and j.
γ	overall external traffic in bps.

3. PROBLEM STATEMENT

We seek to find a quality feasible topology. Since there is no known algorithm that produces a solution of known quality, we rely on design principles to evaluate any network topology. Quality of a topology is measured with respect to three objectives: monetary cost, average network delay per packet (network latency), and maximum number of hops between any source–destination pair. A feasible topology is one that satisfies all design principles. Three categories of 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. High utilization levels cause delays, congestion, and packet loss. Thus the traffic flow on any link i must never exceed a threshold value $\lambda_{max,i}$:

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

where s is the total number of links present in the topology.

(2) The second set of constraints enforces 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 are user-specified and express the designer's desire to enforce certain design guidelines and principles. Examples of guidelines/principles are:

   (*a*) Only specific nodes can be interior nodes of the tree.

   (*b*) Only specific nodes can be directly connected to the backbone.

   (*c*) Certain nodes must be leaf/terminal nodes.

   (*d*) Others.

The stated constraints and design principles define the set of feasible topologies. The search targets feasible topologies which minimize three objectives: monetary cost, average network delay per packet (network packet latency), and maximum hop count for any source–destination pair (tree diameter).

Monetary Cost

The goal is to find a topology with low cost, while meeting all the requirements and constraints. The cost of the cables and the cost of the network devices are the two main entities affecting the monetary cost, *i.e.*,

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

where l represents the total length of cable, c_{cable} represents the cost per unit of the cable used, and c_{nd} *i.e.*, the routers, switches, and hubs used.

Average Network Delay

The second objective is to minimize the average network delay experienced by a packet to go from any source to any destination.

To estimate the average network delay, the aggregate behavior of a link and network device is modeled by an M/M/1 queue [2]. The delay per bit due to the network device feeding the link connecting 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 packet 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 and γ is the sum of all γ_{ij}. The total average network delay is composed of delays of links and network devices and is given by the following equation [2],

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

Maximum Number of Hops Between any Source–Destination Pair

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

4. FUZZY LOGIC

A crisp set is normally defined as a collection of elements $x \in X$, where each element can either belong to a set or not. However, in real life situations, objects do not have crisp [0 or 1] membership criteria. Fuzzy set theory

(FST) aims to represent vague information, like "very hot" and "quite cold", which are difficult to represent in classical (crisp) set theory. In fuzzy sets, an element may partially belong to a set. Formally, a fuzzy set is characterized by a membership function which provides a measure of the degree of presence for every element in the set [19].

Like crisp sets, set operations such as union, intersection, and complementation *etc.*, are also defined on fuzzy sets. There are many operators for fuzzy union and fuzzy intersection. For fuzzy union, the operators are known as s-norm operators while fuzzy intersection operators are known as t-norm. Generally s-norm is implemented using "max" and t-norm as "min" function. However, formulation of multi criteria decision functions do not desire pure "anding" of t-norm nor the pure "oring" of s-norm. The reason for this is the complete lack of compensation of t-norm for any partial fulfillment and complete submission of s-norm to fulfillment of any criteria. Also the indifference to the individual criteria of each of these two forms of operators led to the development of Ordered Weighted Averaging (OWA) operators [20, 21]. This category of operators allows easy adjustment of the degree of "anding" and "oring" embedded in the aggregation. "Orlike" and "Andlike" OWA for two fuzzy sets A and B are implemented as given in Equations (6) and (7) respectively,

$$\mu_{A\cup B}(x) = \beta \times \max(\mu_A, \mu_B) + (1-\beta) \times \frac{1}{2}(\mu_A + \mu_B) \tag{6}$$

$$\mu_{A\cap B}(x) = \beta \times \min(\mu_A, \mu_B) + (1-\beta) \times \frac{1}{2}(\mu_A + \mu_B), \tag{7}$$

where β is a constant parameter in the range [0,1]. It represents the degree to which OWA operator resembles the pure "or" or pure "and" respectively.

4.1. **Fuzzy Reasoning**

Fuzzy reasoning is a mathematical discipline invented to express human reasoning in vigorous mathematical notation. Unlike classical reasoning in which propositions are either true or false, fuzzy logic establishes approximate truth value of propositions based on linguistic variables and inference rules. In order to represent imprecise ideas, Zadeh [19] introduced the concept of linguistic variable. A linguistic variable is a variable whose values are words or sentences in natural or artificial language. The set of values a linguistic value can take is called a term set. This set is constructed by means of primary terms and by placing modifiers known as hedges such as "more", "many", "few" *etc.*, before primary terms. The term set represents a precise syntax in order to form a vast range of values the linguistic variable can take. The linguistic variables can be composed to form propositions using connectors like AND, OR, and NOT.

The quality of a topology is measured against several objective criteria. None of the objectives gives sufficient information to decide the quality of the network topology. Users would like to observe very low latencies, whereas management prefers to have cost as low as possible. In a later section we shall describe how fuzzy logic is used to combine these objectives into a single measure estimating the goodnesses of the movable elements of a solution as well as the membership of a given topology into the fuzzy set of quality topologies.

5. **SIMULATED EVOLUTION**

Simulated Evolution (SE) is a general iterative heuristic proposed by Ralph Kling [22]. It falls in the category of algorithms which emphasize the behavioral link between parents and offspring, or between reproductive populations, rather than the genetic link [23]. This scheme combines iterative improvement and constructive perturbation and saves itself from getting trapped in local minima by following a stochastic perturbation approach. It iteratively operates a sequence of evaluation, selection, and allocation steps on one solution (See Figure 2). The selection and allocation steps constitute a compound move from current solution to another feasible solution

of the state space. SE proceeds as follows. It starts with a randomly or constructively generated valid initial solution. A solution is seen as a set of movable elements. Each element e_i has an associated goodness measure g_i in the interval [0,1]. The main loop of the algorithm consists of three steps: **evaluation**, **selection**, and **allocation**. These steps are carried out repetitively until some stopping condition is satisfied. In the evaluation step, the goodness of each element is estimated. In the selection step, a subset of elements are selected and removed from current solution. The lower the goodness of a particular element, the higher is its selection probability. A bias parameter B is used to compensate for inaccuracies of goodness measure. Finally, the allocation step tries to assign the selected elements to better locations. Other than these three steps, some input parameters for the algorithm are set in an earlier step known as **initialization**.

```
Simulated_Evolution(B, Φ_initial, StoppingCondition)
NOTATION
B= Bias Value.
Φ= Complete solution.
e_i= Individual 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 evaluate g_i;
            /* Only elements that were affected by moves of previous */
            /* iteration get their goodnesses recalculated*/
      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.
   (Simulated_Evolution)
```

Figure 2. Structure of the simulated evolution algorithm.

6. PROPOSED FUZZY SIMULATED EVOLUTION ALGORITHM FOR NETWORK TOPOLOGY DESIGN

In this section, we provide details of each of the phases of the SE algorithm. The pseudocode of the algorithm is given in Figure 2. We confine ourselves to tree design because they are minimal and provide unique paths between every pair of local sites. Further, usually, the design of general network topology starts from a constrained spanning tree.

6.1. Fuzzy Evaluation

The initial spanning tree topology is generated randomly, while taking into account all design constraints mentioned earlier.

The **goodness** of each individual is computed as follows. In our case, an individual is a **link** interconnecting two network devices. 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 [24], Rule 1 translates to the following equation for the fuzzy goodness measure of a link l_i:

$$g_{l_i} = \mu^e(l_i) = \alpha^e \times min(\mu_1^e(l_i), \mu_2^e(l_i)) + (1 - \alpha^e) \times \frac{1}{2} \sum_{i=1}^{2} \mu_i^e(l_i). \tag{8}$$

The superscript e stands for **evaluation** and is used to distinguish similar notation in other fuzzy rules. In Equation 8, $\mu^e(l_i)$ is 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 the 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 resulting function will be a straight line with a negative slope, where the x-axis represents $\frac{LCost}{LCostMax}$, y-axis represents the membership value, $minimum\ cost = \frac{LCostMin}{LCostMax}$, and $maximum\ cost = \frac{LCostMax}{LCostMax} = 1$.

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.* 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 flexibility 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

*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 reasonable limit would be 7, *i.e.*, a branch of the tree should not have more than 7 routers.

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* will have the same shape as in the case of cost membership function above; here the x-axis represents $LDepth$, the y-axis represents the membership value, *minimum depth* = $LDepthMin$, and *maximum depth* = $LDepthMax$.

6.2. **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_{c_i} + B, where B is the selection bias. If $RANDOM > g_{c_i} + B$, 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* [25] 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 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 start of that iteration.

6.3. **Fuzzy Allocation**

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, with the link with the worst goodness being the head-of-line in the queue.

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 and the membership functions for the three criteria are given in Section 6.4 in detail. 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 topologies, as depicted in Figure 3.

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. Of these 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, as shown in Figure 3. 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.

It may so happen that 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 Equation (9) 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. In the allocation phase, we have used tabu search characteristics. As mentioned above, 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 also saved in the *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 of a tree from repetitively removing the same link and replacing it with a link of equal or worse goodness. For details of tabu search, refer to [8].

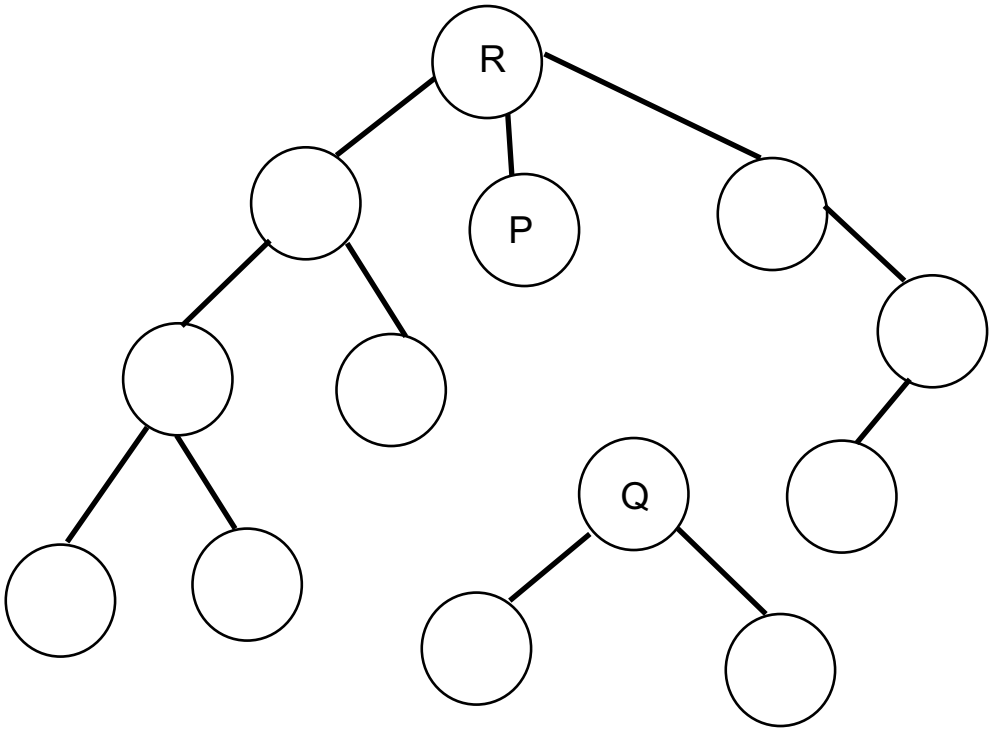

Figure 3. Two disjoint trees containing nodes P *and* Q.

6.4. **Fuzzy Membership Functions**

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 [24], the above fuzzy rule reduces to the following equation:

$$\mu^a(x) = \beta^a \times min(\mu_1^a(x), \mu_2^a(x), \mu_3^a(x)) + (1 - \beta^a) \times \frac{1}{3} \sum_{i=1}^{3} \mu_i^a(x) \tag{9}$$

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 Equation 9 is reported as the best solution found by the SE algorithm.

Below we will see how to get the membership functions for the three criteria we have 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 [7], 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 membership function is again a straight line, where x-axis represents $\frac{TCost}{TCostMax}$, y-axis represents the membership value, $minimum\ cost = \frac{TCostMin}{TCostMax}$, and $maximum\ cost = \frac{TCostMax}{TCostMax} = 1$.

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 Equation (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". In the membership function, x-axis represents $\frac{TDelay}{TDelayMax}$, y-axis represents the membership value, $minimum\ delay = \frac{TDelayMin}{TDelayMax}$, and $maximum\ delay = \frac{TDelayMax}{TDelayMax} = 1$.

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. In the membership function, x-axis represents $THops$, y-axis represents the membership value, $minimum\ hops = THopsMin$, and $maximum\ hops = THopsMax$.

6.5. **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, no significant improvement in solution quality is obtained after the 1^{st} 4000 iterations.

7. 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 each 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 cables 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. For each of the five test cases, ten runs were made to validate the performance of the proposed algorithm, where each run started with a different initial solution.

Table 1. Characteristics of Test Cases Used in Our Experiments.

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

LCostMin, LCostMax, and TCostMin are in US$. TDelayMin is in Milliseconds. Traffic is in Mbps.

7.1. Comparison of Simulated Evolution and Simulated Annealing

We compared the proposed SE with simulated annealing (SA) algorithm [8]. SA has four important parameters which need to be tuned very carefully. These are: cooling rate α, constant β, initial temperature T_0, and M which represents the time until next parameter update. After trial runs, appropriate values of these parameters were found to be $\alpha = 0.9$, $\beta = 1.0$, $T_0 = 10$, and $M = 10$.

Table 2 presents the results for SA and SE. From this table, it is observed that SE performs better than SA as far as monetary cost objective is concerned. In all the test cases, better solutions are achieved by SE. For example, a gain of 12.6 % is achieved in case of **n50**. A similar behavior is seen for average network delay metric, where SE achieves gain in all the cases, *e.g.* in case of **n40**, where a gain of 14.81% is observed. Similarly, for maximum number of hops metric, a gain is achieved for small (**n15**) and medium (**n33**) size test cases, with a loss of one hop for **n40** and **n50**. However, the loss in maximum hops for **n40** and **n50** is compensated by the improvement in the monetary cost and average network delay metrics. As far as the execution time is concerned, SE has a slightly higher execution time than SA in most of the cases. It is due to the fact that SA performs one move per iteration while SE performs multiple moves in a single iteration.

In order to compare the quality of search space of SA and SE, frequency of solutions for different membership ranges is plotted against the membership value in Figure 4(a) for **n25**. In SA, only one individual (link) is selected at a time, and only one move is allowed for that selected link. If the new link is not a feasible one or does not pass the Metropolis criterion [8], then the original link is placed back in its position. This implies that there may be iterations where original links are placed back and no alteration takes place to the currently existing solution. Therefore, such iterations are not included as giving a new solution for SA. This in turn gives a total of 2200 iterations (out of 4000 iterations) where an alteration took place in the solution. Thus, only these 2200 iterations (solutions) are plotted in Figure 4(a). In this figure, it is observed that SE has more solutions falling in higher membership ranges than SA, suggesting that SE has investigated a better solution subspace. For example, SA has most of the solutions in the membership range 0.2–0.25, whereas SE has most of the solutions in the range 0.3–0.35.

Table 3 provides the variance, maximum, and average of membership function "Good Solution" for the ten runs for SE. From these values, we observe that the final solutions are of the same quality, even if they start with different initial solutions.

Table 2. Comparison of SA and SE.

Case	SA				SE					% Gain		
	C	D	H	T	TL	C	D	H	T	C	D	H
$n15$	318000	3.469	6	1.17	2	297100	2.78	4	2.25	6.57	19.86	33.3
$n25$	511320	3.725	6	3	5	483210	3.537	6	4	5.497	5.047	0
$n33$	708135	5.189	9	5.5	6	682465	4.19	6	8	2.76	19.25	33.3
$n40$	903735	5.213	8	27.5	7	783970	4.441	9	26	13.25	14.81	−11.1
$n50$	1124720	5.943	10	57	7	983020	5.245	11	65	12.6	11.74	−9.09

C = Cost in US $, D = Delay in Milliseconds per Packet, H = hops, T = Execution Time in Minutes, TL= Tabu List Size.
The percentage gain shows the improvement achieved by SE compared to SA.

Table 3. Variance, Maximum, and Average Membership Values for Fuzzy Function "Good Solution" for SE. The values in boldface represent the memberships for best solutions reported in Table 2.

	$n15$	$n25$	$n33$	$n40$	$n50$
	0.302105	0.351167	0.313853	0.277301	0.314923
	0.318251	0.343485	0.295764	0.319078	0.30057
	0.296676	0.344844	0.318207	0.329939	0.312573
	0.298715	0.305375	**0.352941**	0.312572	0.291446
	0.30452	0.349154	0.323703	**0.337554**	0.31793
	0.28889	**0.352056**	0.347812	0.303501	**0.32212**
	0.312287	0.332685	0.308764	0.337417	0.30118
	0.27984	0.31270	0.338931	0.327757	0.297819
	0.30562	0.3	0.310076	0.3019	0.27434
	0.25557	0.34751	0.296702	0.294715	0.258745
Variance	0.00032403	0.000408	0.000402	0.000394	0.000403
Maximum	0.312851	0.352056	0.352941	0.337554	0.32212
Average	0.296247	0.333898	0.320675	0.314173	0.299165

Figures 4(b) and (c) illustrate very interesting and desirable behavior of SE. The figures plot the cumulative frequency of solutions found at different membership ranges after each 200 iterations for **n25**. In Figure 4(b), the plot for SA shows that most of the solutions fall in the membership range of 0.2–0.25 where the solutions in this range are few initially but with the passage of iterations, the cumulative frequency of solutions increases. This figure also shows that there are no solutions at all in higher membership ranges. On the other hand, the cumulative distribution of solutions in different ranges for SE in Figure 4(c) shows a pattern where solutions with higher membership ranges are initially less. As more iterations are executed, the cumulative frequency of solutions with higher membership ranges increases, while the cumulative frequency of solution with lower membership ranges remains the same, suggesting that with time, the algorithm is evolving towards better solution subspaces.

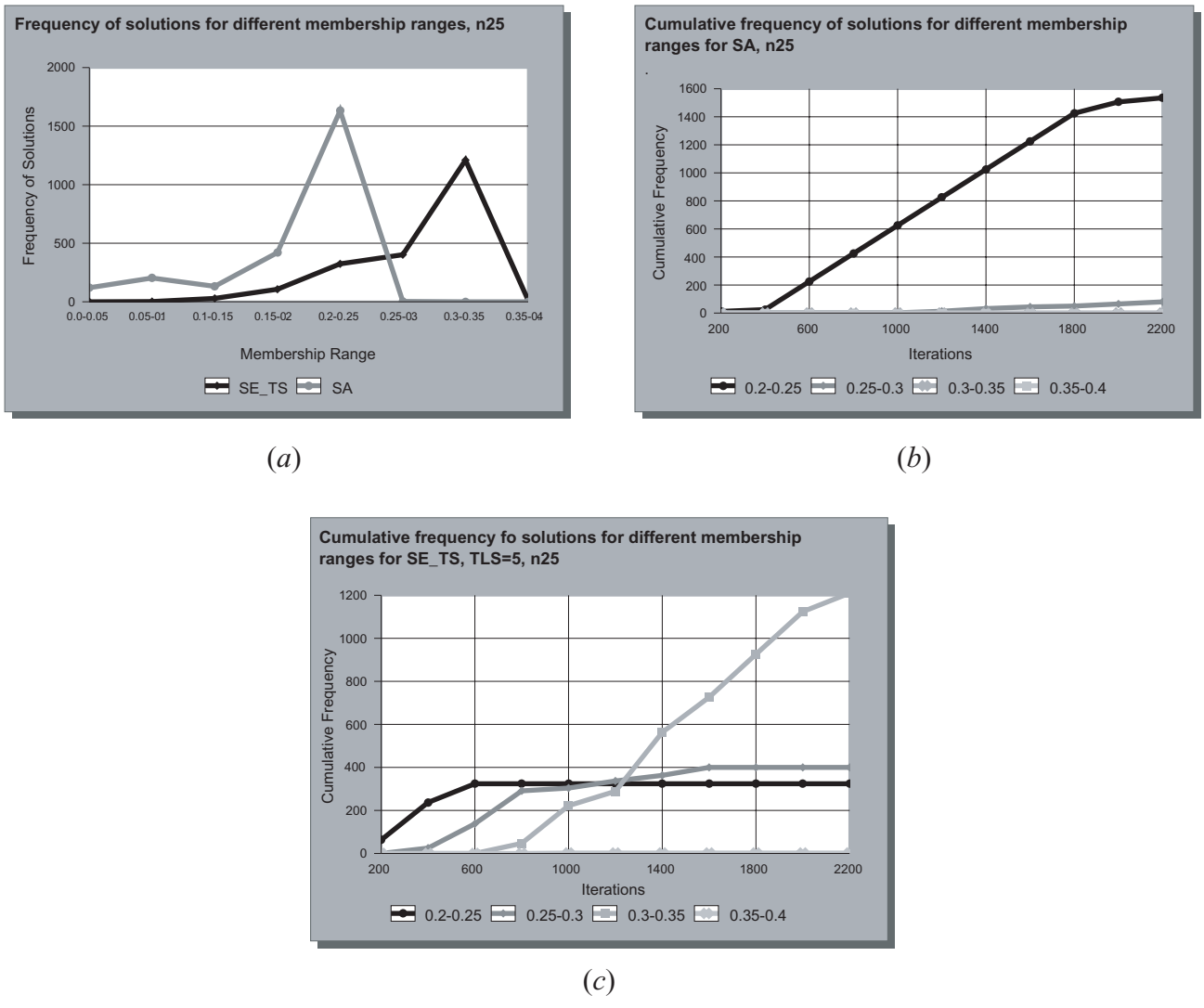

Figure 4. Comparison of SE algorithm with SA for different membership ranges in fuzzy set "good topology". (a), (b), *and* (c) *respectively compare the frequency of solutions, cumulative frequency of solutions for SA, and cumulative frequency of solutions for SE.*

7.2. **Comparison of SE and Esau–Williams Algorithm**

The proposed SE algorithm was also compared with the Esau–Williams (EW) algorithm, which is one of the most widely used constructive algorithm for centralized network design. The EW algorithm was adapted to meet the constraints and objective requirements. Table 4 presents the results for EW algorithm and best tabu list size SE. From this table it is quite clear that SE has far better overall performance than EW algorithm. We observe that although SE has a little inferior performance than EW with respect to monetary cost, yet if we consider the other two objectives, we see a significant improvement by SE. The inferiority in monetary cost and superiority in average delay and maximum hops of SE is due to the fact that EW algorithm is considering only the cost as the objective to be optimized, therefore it reduces only the cost, ignoring the other two objectives. We also see that although EW algorithm is optimizing only the cost, it does not have a very significant improvement than SE with respect to monetary cost objective. On the other hand, SE has a significant gain in delay and hops objectives. Furthermore, EW algorithm has a local partial view of the search and always makes a move greedily, while SE works always with complete solution and accepts bad moves to get out of local minima. The difference in execution times of the two algorithms is due to the reason that EW algorithm is a constructive heuristic, while SE is an iterative heuristic.

Table 4. Comparison of EW and SE_TS.

Case	EW				SE_TS					% Gain		
	C	D	H	T	TL	C	D	H	T	C	D	H
$n15$	292700	5.0012	9	0.017	2	297100	2.78	4	2.25	−1.48	44.4	55.56
$n25$	469790	5.002	9	0.033	5	483210	3.537	6	4	−2.78	29.29	33.3
$n33$	624180	5.642	10	0.033	6	682465	4.19	6	8	−8.54	25.74	40
$n40$	754445	13.491	14	0.033	7	783970	4.441	9	26	−3.77	67.1	35.71
$n50$	928105	7.167	14	0.083	7	983020	5.245	11	65	−5.58	26.82	21.42

8. **CONCLUSION**

In this paper we have presented a fuzzy SE for backbone topology design of campus networks. The proposed SE algorithm was always able to find feasible topologies with desirable qualities. Comparison with SA showed that the search performed by SE is more intelligent, that is, the solution subspace investigated by SE is of superior quality than that of SA. Further, as time elapsed, SE progressively evolved toward better solutions, a desirable characteristic of evolutionary heuristics. Results also suggest that fuzzy simulated evolution algorithm performs better than the Esau–Williams algorithm which is a widely used algorithm for centralized network design.

ACKNOWLEDGMENTS

The 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, Dhahran, Saudi Arabia*, October 1997, pp. 1–18.

[2] R. Elbaum and M. Sidi, "Topological Design of Local-Area Networks Using Genetic Algorithm", in *IEEE/ACM Transactions on Networking*, October 1996, pp. 766–778.

[3] M. Gen, K. Ida, and J. Kim, "A Spanning Tree-Based Genetic Algorithm for Bicriteria Topological Network Design", in *IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, U.S.A.*, May 1998, pp. 164–173.

[4] C. Ersoy and S. Panwar, "Topological Design of Interconnected LAN/MAN Networks", *IEEE Journal on Selected Area in Communications*, October 1993, pp. 1172–1182.

[5] Dimitri Bertsekas and Robert Gallager, *Data Networks*, 2nd edn. New York: Prentice-Hall Inc., 1992.

[6] R.C. Prim, "Shortest Connection Networks and Some Generalizations", *BSTJ*, **36** (1957), pp. 1389–1401.

[7] L.R. Esau and K.C Williams, "On Teleprocessing System Design. A Method for Approximating the Optimal Network", *IBM System Journal*, **5** (1966), pp. 142–147.

[8] Sadiq M. Sait and Habib Youssef, *Iterative Computer Algorithms and their Application to Engineering*. Palo Alto, California, U.S.A.: IEEE Computer Science Press, December 1999.

[9] S. Kirkpatrick, C. Gelatt Jr., and M.Vecchi, "Optimization by Simulated Annealing", *Science*, May 1983, pp. 498–516.

[10] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley Publishing Company, 1989.

[11] B. Dengiz, F. Altiparmak, and A. Smith, "Local Search Genetic Algorithm for Optimal Design of Reliable Network", *IEEE Transactions on Evolutionary Computation*, September 1997, pp. 179–188.

[12] S. Pierre and G. Legault, "A Genetic Algorithm for Designing Distributed Computer Network Topologies", *IEEE Transactions on Systems, Man, Cybernetics*, **28**(**2**) (1998), pp. 249–258.

[13] B. Dengiz, F. Altiparmak, and A. Smith, "Efficient Optimization of All-Terminal Reliable Networks, Using an Evolutionary Approach", *IEEE Transactions on Reliability*, March 1997, pp. 18–25.

[14] Peter C. Fetterolf, "Design of Data Networks with Spanning Tree Bridges", *IEEE International Conference on Systems, Man, and Cybernetics, Los Angeles, California, U.S.A.*, 1990, pp. 298–300.

[15] S. Pierre and A. Elgibaoui, "A Tabu Search Approach for Designing Computer Network Topologies with Unreliable Components", *IEEE Transactions on Reliability*, **46**(**3**) (1997), pp. 350–359.

[16] Salman A. Khan, "Topology Design of Enterprise Networks", *MS Thesis, King Fahd University of Petroleum and Minerals*, 1999.

[17] Mischa Shwartz, *Telecommunications Networks: Protocols, Modeling, and Analysis*. New York: Addison-Wesley Publishing Company, 1987.

[18] Gerd E. Keiser, *Local Area Networks*. New York: McGraw-Hill Book Company, 1989.

[19] L.A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes", *IEEE Transactions on Systems, Man, Cybernetics*, **SMC-3**(**1**) (1973), pp. 28–44.

[20] R. Yager, "Multiple Objective Decision-Making Using Fuzzy Sets", *International Journal of Man–Machine Studies*, **9** (1977), pp. 375–382.

[21] R. Yager, "Second Order Structures in Multi-Criteria Decision Making", *International Journal of Man–Machine Studies*, **36** (1992), pp. 553–570.

[22] Ralph Michael Kling, "Optimization by Simulated Evolution and Its Application to Cell Placement", *Ph.D. Thesis, University of Illinois, Urbana*, 1990.

[23] D.B. Fogel, "An Introduction to Simulated Evolutionary Optimization", *IEEE Transactions on Neural Networks*, **5**(**1**) (1994), pp. 3–14.

[24] Ronald Y. Yager, "On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking", *IEEE Transactions on Systems, Man, and Cybernetics*, **18**(**1**) (1988), pp. 183–190.

[25] Sadiq M. Sait, Habib Youssef, and Ali Hussain, "Fuzzy Simulated Evolution Algorithm for Multiobjective Optimization of VLSI Placement", *IEEE Congress on Evolutionary Computation, Washington, D.C., U.S.A.*, July 1999, pp. 91–97.

Paper Received 14 January 2001; Revised 9 November 2002; Accepted 21 January 2003.