Available online at www.sciencedirect.com
SCIENCE DIRECT®
PERGAMON
Engineering Applications of Artificial Intelligence 15 (2002) 327–340
Engineering Applications of
ARTIFICIAL
INTELLIGENCE
www.elsevier.com/locate/engappai

Topology design of switched enterprise networks using a fuzzy simulated evolution algorithm

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

Department of Computer Engineering, King Fahd University of Petroleum and Minerals, KFUPM Box 6573, Dhahran-31261, Saudi Arabia

Abstract

The topology design of switched enterprise networks (SENs) is a hard constrained combinatorial optimization problem. The problem consists of deciding the number, types, and locations of the network active elements (hubs, switches, and routers), as well as the links and their capacities. Several conflicting objectives such as monetary cost, network delay, and maximum number of hops have to be optimized to achieve a desirable solution. 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 an approach based on Simulated Evolution algorithm for the design of SEN topology. The overall cost function has been developed using fuzzy logic. Several variants of the algorithm are proposed and compared together via simulation and experimental results are provided.

Keywords: Enterprise networks; Simulated evolution; Fuzzy logic; NP-hard; Multiobjective optimization

1. Introduction

In an enterprise network, a large number of *nodes* are interconnected together through a computer network. These nodes are divided into two classes: *end-user nodes* and *network active elements*. End-user nodes represent access points such as workstations, personal computers, printers, mainframe computers, etc. Network active elements consist of 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. Several constraints dictate the network topology. The internetworks can be broken down into smaller parts or group of nodes, as governed by geographical constraints. Each group makes up a local area network (LAN) (Ersoy and Panwar, 1993). According to Open System Interconnection (OSI) Reference Model, there are seven "layers" which are used in communication between two nodes. Each layer has certain functions to perform and has its own characteristics. 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 enterprise network, which consists of a backbone interconnecting a number of LANs via routers or layer 3 switches (see Fig. 1). Further, the nodes of a LAN may be subdivided into smaller parts, called *LAN segments*. The hierarchical structuring of the network into a backbone, interconnecting a number of LANs via routers/layer 3 switches, where each LAN is an interconnected collection of LAN segments, achieves several desirable effects:

1. It confines broadcast traffic to a single LAN, thus avoiding the problem of having broadcast storms sweeping across the entire network.
2. Switched structured topologies ensure highest network availability, lowest latency, and provide the most appropriate connectivity to users.
3. Reduces cost of administration. Equipment moves and changes are carried out in a more orderly fashion. Further, diagnosis and correction of network problems are easier.

The topology design of LAN itself consists of two main issues: *segmentation*, where LAN segments are found, and *design of actual topology*, which consists of interconnecting the individual segments. In recent years, *Switched LANs* have gained popularity since they follow modern recommended structured cabling standard.

*Corresponding author. Tel.: +966-3-860-4268; fax: +966-3-860-3059.
E-mail address: youssef@ccse.kfupm.edu.sa (H. Youssef).

0952-1976/02/$ - see front matter © 2002 Elsevier Science Ltd. All rights reserved.
PII: S0952-1976(02)00066-0

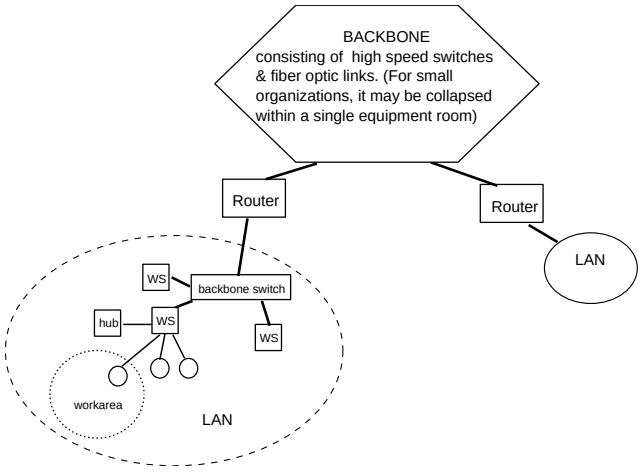

Fig. 1. A typical enterprise network (WS represents workgroup switch).

These LANs follow a hierarchical star topology and use switching technology to facilitate increased bandwidth at the desktop and backbone necessary for network-hungry applications (Black, 1998). Traffic segmentation becomes easy by switching. This provides discrete LAN interfaces that are connected to form a bigger LAN (Black, 1998). Switches can switch all the current technologies: Ethernet, Fast Ethernet, Gigabit Ethernet, Token Ring, ATM, and FDDI. A big advantage of switched networks is that central switch upgrades can be done easily. This is done to increase performance and is transparent to the user. For example, if a user has 10/100 Mbps NIC and Category 5 cabling, a central change from a 10 Mbps port to a 100 Mbps port will increase the user's bandwidth by a factor of 10 without even going to the user's office (Black, 1998).

Switched LANs consist of the following general hierarchy of components (in order of power and price, from highest to lowest) (Black, 1998):

- *Routers*: Provide most effective isolation among LANs. Most often, routing technology is integrated into backbone or workgroup switches.
- *Backbone switches*: These high-end switches are deployed at the core of a network and use switching technology. These switches aggregate data from hubs and workgroup switches providing interconnection among these devices. Backbone switches typically accept various cards with different technologies.
- *Workgroup switches*: These lower-end network devices aggregate multiple shared segments by using switching technology. Often a workgroup switch switches onto a high-speed backbone connection such as Fast Ethernet, FDDI, Gigabit Ethernet, or even ATM.
- *Hubs*: These can be subdivided into two types: *chassis hubs* and *stackable hubs*. Chassis hubs contain a variety of network modules and high-speed backplane capable of housing repeaters, bridges, switches, or concentrators. Stackable hubs provide shared media access by logically extending the backplane. The backplane is a shared bus across the stackable units over which data can be transmitted.

Thus, the design of such a switched enterprise network can be approached in four steps:

1. Assignment of users/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).
4. Backbone design, where the local sites are connected to the backbone.

Topological design of SENs is a hard problem (Youssef et al., 1997). A class of heuristics that have been found effective for this category of problems are iterative heuristics such as simulated annealing (SA), simulated evolution (SE), genetic algorithm (GA), and tabu search (TS). These heuristics, when properly engineered, are always able to find near optimal solutions, regardless of the initial solution at which they start the search from. In this work, we propose a hybrid meta-heuristic for the topology design problem which follows the search strategy of SE algorithm.

1.1. Literature review and related work

Topological design of enterprise networks is a hard problem (Youssef et al., 1997). Even the design of a LAN is itself an NP-hard problem (Ersoy and Panwar, 1993; Elbaum and Sidi, 1996; Gen et al., 1998). The state space is of exponential complexity. For example, for a network with n nodes, there can be as many as $2^{n(n-1)/2}$ different topologies. Even for $n = 20$ this number evaluates to more than 10^{56}. Therefore, in order to produce good feasible solutions in a reasonable amount of time, approximation methods known as 'heuristics' are used to focus the search on feasible topologies of desirable characteristics.

There are two categories of heuristics: *constructive* and *iterative*. Constructive schemes build a topology in a piecewise manner. These schemes join two nodes together at a time, until the topology is complete. Such schemes are fast but fall short of good topology. This is due to the fact that these schemes make decision about joining two nodes on the basis of partial topology only. Secondly, once a decision is made, no matter how bad it is, there is no mechanism to reverse it. Therefore, these schemes may remain trapped in local minima. Some of the well known constructive algorithms for the constrained minimum spanning tree problem are Kruskal

algorithm (Bertsekas and Gallager, 1992), Prim algorithm (Prim 1957), and Esau and Williams (1966) algorithm.

In contrast to constructive techniques, iterative schemes require larger computational time to generate good network topologies. Iterative heuristics attempt to improve a complete solution by making controlled walk through the state space (Sait and Youssef, 1999). Iterative techniques start with an initial solution and repeatedly modify the solution in each iteration until no more improvement occurs. The modification in solution is intended to reduce the cost of the solution. Iterative schemes can be further classified on the basis of whether they can accept bad solutions probabilistically or not. Those iterative algorithms which allow acceptance of bad moves probabilistically fall in the sub-category of "stochastic iterative algorithms". This property is called "hill climbing property". It saves algorithms from getting trapped in local minima. However, it is required that acceptance of bad moves be controlled to avoid random traversal of search space. Examples of such probabilistic iterative schemes are SA (Kirkpatrick et al., 1983), GA (Goldberg, 1989). SE and TS (Sait and Youssef, 1999).

The use of iterative heuristics for topological network design has been reported in many research papers. Simulated annealing for network topology design has been reported in (Ersoy and Panwar (1993)) 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 (1990) used simulated annealing to design LAN–WAN computer networks with transparent bridges. He developed mathematical models of LAN–WAN networks and formulated an optimization problem. A simulated annealing algorithm was proposed which generates sequences of neighboring spanning trees and evaluates design constraints based on maximum flow, bridge capacity, and end-to-end delay. As the annealing temperature is lowered, the algorithm moves towards the global optimal solution. Experimental results showed that LAN–WAN designs using simulated annealing were better than 99.99% of all feasible designs.

Similarly, GA for topological network design has been proposed in Elbaum and Sidi (1996); Gen et al. (1998); Dengiz et al. (1997a,b); and Pierre and Legault (1998). Pierre and Legault (1998) used 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 and Sidi (1996) have used GA for designing LANs with the objective of minimizing the average network delay. The constraints they considered is that the flow on any link does not exceed the capacity of that link. The topology design issues they addressed consists 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. (1998) 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. (1997a,b) 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).

Many combinatorial optimization problems can be formulated as follows: *Given a finite set E 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 $E = \{e_1, e_2, \ldots, e_n\}$ and a set of locations $L = \{0, 1\}$, where $L(e_i) = 1$ if link e_i belongs to the topology and $L(e_i) = 0$ otherwise.

In this work, a SE-based heuristic (Sait and Youssef, 1999) is used for topology design of structured enterprise 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 (Khan, 1999).

For enterprise 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.

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 golas, 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 are implemented using fuzzy algebra (Zadeh, 1965).

1.2. SE algorithm

Simulated evolution is a stochastic evolutionary search strategy that falls in the general category of meta-heuristics. It was first proposed by Kling and Banerjee (1989). SE adopts the generic state model described above, where a solution is seen as a population of movable elements. Each element i is characterized by a **goodness** measure $g_i \in [0,1]$ where $g_i = O_i/C_i$. C_i is the estimated real cost of element e_i in its position in current state, and O_i is a lower bound on the cost of e_i. For example, in the case of network optimization problem, the cost of a given link e_i could be taken as the utilization of that link. For each link e_i, g_i can be estimated by taking the ratio of the current utilization of the link to the optimum utilization of the link, which could be found through some mathematical approach.

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 pseudocode 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 Kling and Banerjee (1989).

Elements selected during the **selection** step are assigned to new locations in the **allocation** step with

```
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.

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 and must include domain-specific knowledge. Different constructive allocation schemes are proposed in Kling and Banerjee (1989).

Though SE falls in the category of meta-heuristics such as simulated annealing and genetic algorithm GA, there are significant differences between these heuristics (see Sait and Youssef, 1999). A classification of meta-heuristics proposed by Glover and Laguna (1997) 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 $A/S/1$. Since SE is a memoryless meta-heuristic, 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 TS approach.

1.3. 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 (Zadeh, 1965). 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 modelled to include multiple objectives. These phases are **evaluation** and **allocation**. We have used fuzzy logic-based reasoning in these two phases, details of which are covered in the following section.

2. 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 section 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

3. Problem statement

Our objective is to find the feasible topology of near optimum *overall cost*. A feasible topology is one that satisfies design constraints. Optimality 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. However, it is important to mention that increasing the number of conflicting objectives will not have influence on the effectiveness on the proposed SE algorithm.

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:
$$\lambda_i < \lambda_{\max,i} \quad i = 1, 2, \ldots, s, \tag{1}$$
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.
$$\sum_{j=1}^{n} t_{ij} < P_i \quad i = 1, 2, \ldots, n \quad \forall i \neq j \tag{2}$$
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.

3.1. 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

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

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.

3.2. 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 Elbaum and Sidi (1996). The delay per bit due to the network device between local sites i and j is $B_{i,j} = \mu b_{i,j}$, where $1/\mu$ 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 (Elbaum and Sidi, 1996)

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

3.3. 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.

4. 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.

4.1. Initialization

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

4.2. 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 (Yager, 1998), Rule 1 translates to the following equation for the fuzzy goodness measure of a link l_i.

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

The superscript e stands for **evaluation** and is used to distinguish similar notation in other fuzzy rules. In (6), $\mu^{\text{e}}(l_i)$ is the membership in the fuzzy set of *high goodness links* and α^{e} is a constant. The $\mu_1^{\text{e}}(l_i)$ 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

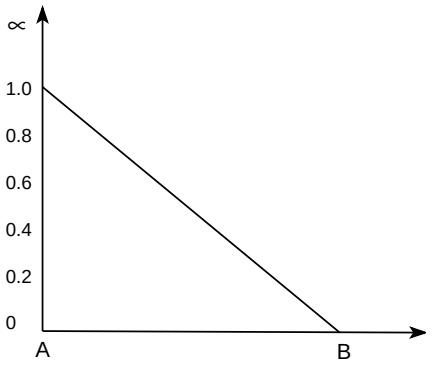

Fig. 3. Membership function for the objective to be optimized.

as depicted in Fig. 3, where x-axis represents LCost/LCostMax, y-axis represents the membership value, A = LCostMin/LCostMax, and B = LCostMax/LCostMax = 1. This normalization enables us to use the same membership function for all topology design instances.

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 that 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.

[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 seven routers.

4.3. Selection

In this stage of the algorithm, for each link l_i in current tree topology, where $i = 1, 2, \ldots, n-1$, 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* (Youssef et al., 2001) 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:

$$B_k = 1 - G_{k-1},$$

where B_k is the bias for kth iteration and G_{k-1} is average goodness of all the links at the end of $(k-1)$st iteration.

4.3.1. 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.

In the *fuzzy allocation scheme*, the three criteria to be optimized are combined using fuzzy logic to characterize a good topology, as depicted in Fig. 4.

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.

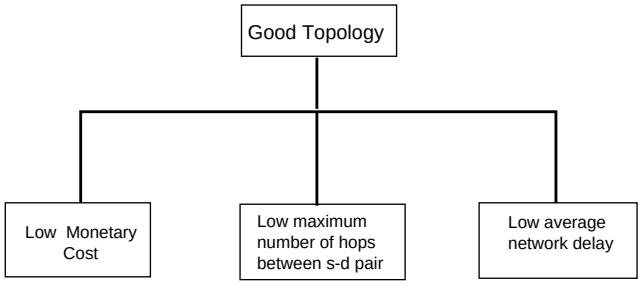

Fig. 4. Basic components for a good topology.

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 (Yager, 1998), the above fuzzy rule reduces to the following equation.

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

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 Eq. (7) 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 mentioned above.

4.3.2. 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 (1966), 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 TCost/TCostMax, y-axis represents the membership value, A = TCostMin/TCostMax, and B = TCostMax/TCostMax = 1.

4.3.3. 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 Eq. (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 TDelay/TDelayMax, y-axis represents the membership value, A = TDelayMin/TDelayMax, and B = TDelayMax/TDelayMax = 1.

4.3.4. 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 topologies, as depicted in Fig. 5.

Now the placing of trial links begins. In this work, the approach to place trial links is as follows. At most 10 trial moves (i.e., trial links) are evaluated for each removed link. One point to mention is that for the 10 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 10 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 10 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

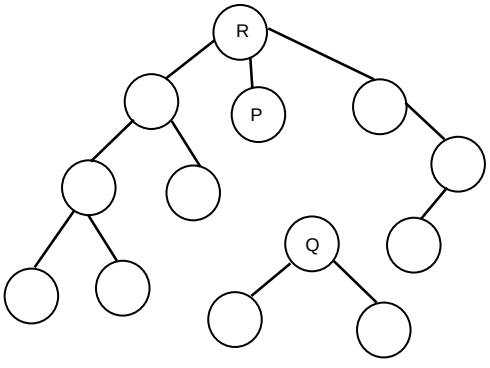

Fig. 5. Two disjoint trees containing nodes P and Q.

the random moves, we select any two nodes in the two subtrees and connect them.

It may so happen that all the 10 moves are invalid, in which case the original link is placed back in its position. The valid moves are evaluated based on Eq. (7) and the best move among the 10 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 in this paper. The first one is the same as has been described above, which we call SE. In the second variation, TS characteristics have been introduced, details of which follow.

4.4. TS based Allocation

Tabu search (TS) is a general iterative heuristic that is used for solving combinatorial optimization problems. The algorithm was first presented by Glover and Laguna (1997). 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 (Glover and Laguna, 1997). It searches for the best move in the neighborhood of the current solution. As opposed to local search, TS does not get trapped in local optima because it accepts bad moves if they are expected to lead to unvisited solutions.

In its very basic operation, TS works as follows. It starts with an initial solution s that is selected randomly or using any constructive algorithm. It defines a subset $V*(s)$ of its neighborhood $N(s)$. The algorithm evaluates the solution in $V*(s)$ and finds the best (in terms of evaluation function) among them, call it s^* to be considered as the next solution. The algorithm maintains a list of *attributes* of accepted moves for the purpose of preventing cycling back to recently visited solutions. This list is called *Tabu List*. The size of tabu list specifies the number of coming iterations for which the move remains tabu. An *attribute* is some characteristic of a move which is saved in the tabu list, since it is not feasible to store the whole solution when the solution representation is large or complex. If the tabu list does not define the move leading to s^* as tabu, it is accepted as the new solution even if it is worse than the current solution in terms of the evaluation function. However, if the move leading to s^* is defined as tabu by the tabu list, the solution is not accepted until it has one or more features that makes the algorithm override its tabu status to accept it. *Aspiration criterion* is used to check whether the tabu solution is to be accepted or not. For more details, interested readers are referred to (Sait and Youssef, 1999; Glover and Laguna, 1997).

In this work, we have modified the SE algorithm by introducing TS 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 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.

4.5. 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, the SE algorithm converges within 4000 iterations or less.

5. Results and discussion

This section summarizes the experimental study of variations of SE algorithms implemented in this work, namely

1. Fuzzy evaluation and fuzzy allocation based simulated evolution algorithm implementation with fixed bias controlled selection. This implementation is labelled as SE_FF.
2. Fuzzy evaluation and fuzzy allocation based simulated evolution algorithm implementation with

Table 1
Classification of our SE implementations

Algorithm	Evaluation	Selection	Allocation
SE_FF	Fuzzy (link cost, depth)	Fixed bias	Fuzzy (cost, delay, hops)
SE_VB	Fuzzy (link cost, depth)	Variable bias	Fuzzy (cost, delay, hops)
SE_TS	Fuzzy (link cost, depth)	Variable bias	Fuzzy (cost, delay, hops) with tabu search approach

Table 2
Parameter values for different stages of the SE algorithm

Stage	Parameters
Fuzzy allocation	$\beta^a = 0.5$
Fuzzy evaluation	$\beta^e = 0.5$
Stopping condition	Fixed 4000 iterations
Initial solution	Random
Bias (B)	Fixed, variable

Table 3
Characteristics of test cases used in our experiments. LCostMin, LCostMax, and TCostMin are in dollars. TDelayMin is in milliseconds. Traffic is in Mbps

Name	No. of Local sites	LCostMin	LCostMax	TCostMin	TDelayMin	Traffic
*n*15	15	1100	9400	325,400	2.14296	24.63
*n*25	25	530	8655	469,790	2.15059	74.12
*n*33	33	600	10,925	624,180	2.15444	117.81
*n*40	40	600	11,560	754,445	2.08757	144.76
*n*50	50	600	13,840	928,105	2.08965	164.12

Table 4
Equipment parameters and their characteristics assumed in our experiments

Parameter	Characteristic
Cost of backbone switch with FO interface (eight ports)	$ 30,000
Cost of router with FO interface (four ports)	$ 20,000
Cost of switch with FO interface (eight ports)	$ 15,000
Cost of hub or MAU with FO interface (12 ports)	$ 5,000
Cost of fiber optic cable	$ 5/m
Cost of Category 5 UTP	$ 0.5/m
Delay per bit due to forwarding device	250 μ/s.
Maximum traffic on a link allowed	60%
Average packet size	500 bytes

variable bias controlled selection. This implementation is labelled as SE_VB.

3. Fuzzy evaluation and fuzzy allocation based simulated evolution algorithm implementation where the allocation scheme incorporates tabu search characteristics. This implementation has variable bias controlled selection and is identified as SE_TS.

The characteristics of above listed algorithm variations are summarized in Table 1. The parameter values for different stages of the SE algorithm are summarized in Table 2. For example, the values of α^e and β^a were chosen after several trials of combinations of these.

We tested our implementations of the simulated evolution algorithm on five arbitrary test cases. Ten experiments were run for each test case to validate and compare the performance of each SE implementation method described above. 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 links were obtained from vendors. The characteristics of test cases are listed in Table 3. The smallest test network has 15 local sites and the largest has 50 local sites. Also, Table 4 lists the characteristics of the equipment used in our experiments.

Table 5
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
	1	298,200	2.935	4
	2	297,100	2.78	4
*n*15	3	294,350	3.448	6
	4	298,100	3.037	5
	5	296,900	3.278	6
	3	481,745	4.219	8
	4	478,690	4.189	9
*n*25	5	483,210	3.537	6
	6	479,915	4.275	9
	7	488,400	4.608	9
	3	655,715	5.772	11
	5	652,785	4.77	8
*n*33	6	682,465	4.19	6
	7	652,310	5.95	10
	9	667,100	5.087	7
	5	785,795	4.746	10
	6	798,695	8.019	12
*n*40	7	783,970	4.441	9
	8	786,950	5.478	9
	9	790,645	5.136	8
	4	958,995	6.739	14
	5	967,110	9.279	14
*n*50	7	983,020	5.245	11
	8	1075,450	5.725	9
	9	971,965	7.13	12

Following sets of experiments were carried out.

1. Analyzing the effect of TS approach based allocation in SE_TS and understanding the effect of tabu list size.
2. Comparing SE_FF, SE_VB, and SE_TS.

5.1. Effect of TS based allocation and tabu list size

Table 5 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.

In order to see the behavior of our SE_TS algorithm, we plot different parameters versus iteration count for the test case *n50* (tabu list size = 7). Fig. 6(a) shows the best monetary cost of solutions found. A similar behavior is seen in Fig. 6(b) and (c) which, respectively, plot the best values of average network delay and maximum number of hops. The stability of the algorithm becomes more obvious in Fig. 6(d) where the best membership in fuzzy function "good topology" is plotted against iterations. The figure shows small moves with constant convergence towards the optimum. Fig. 6(e) shows the frequency of solutions having membership values in different ranges. According to this figure, out of 4000 solutions (since we run the algorithm for 4000 iterations), most of the solutions are falling in higher membership ranges, suggesting that the algorithm is concentrating in good solutions subspace. The convergence of the algorithm towards better solution is dependent on the average goodness of links. A high average goodness of links suggests that better links have been found and that there is a little room for further improvement. This is seen from Fig. 6(f), where we see that the average goodness of links is increasing with each iteration. This improvement slows down towards the end, suggesting that better links have been found throughout the topology.

Table 6 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

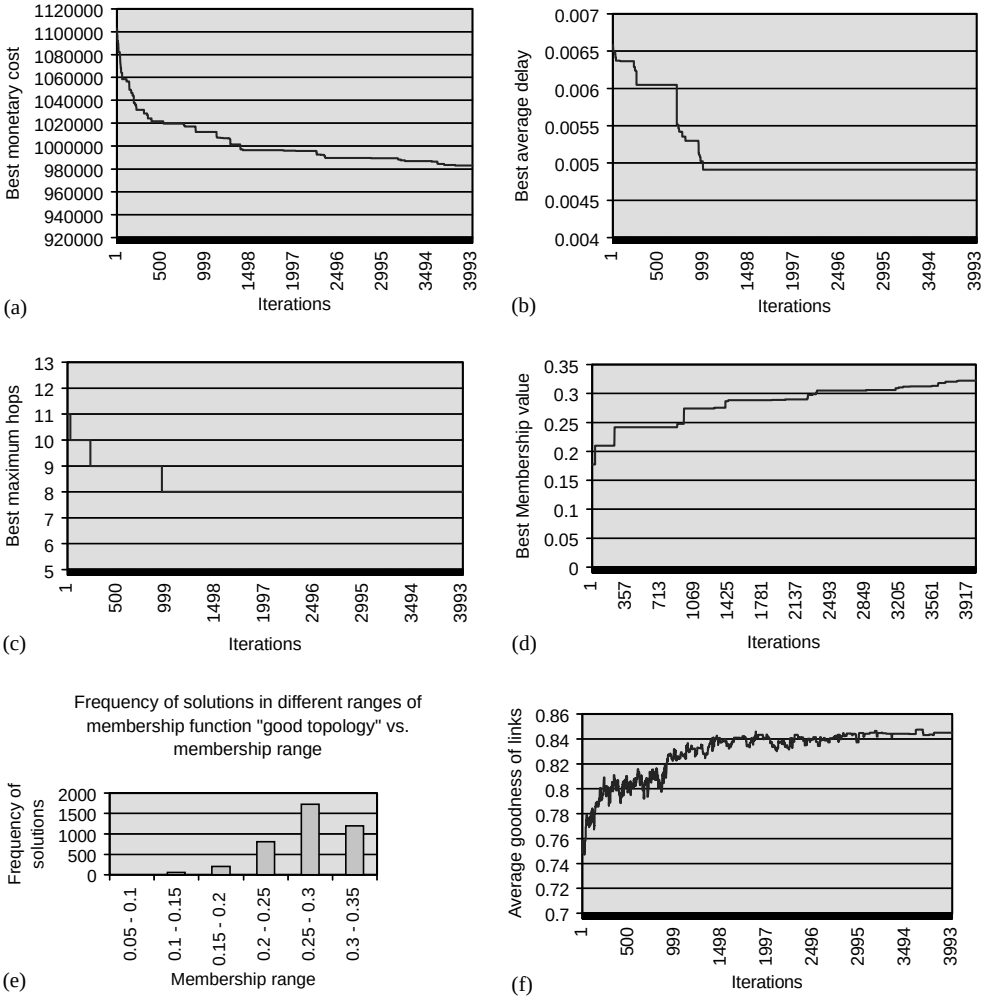

Fig. 6. Plots for different parameters for *n*50, tabu list size = 7: (a) best monetary cost; (b) best average network delay; (c) best maximum number of hops; (d) best membership value in "Good Topology"; (e) frequency of solutions in different membership ranges and (f) average goodness of links.

Table 6
Results for best tabu list size. Execution time is in minutes

Test case	Tabu list size for best solution	Total moves	Tabu moves	% of Tabu moves	Exec. time
*n*15	2	1241	45	3.62	2.25
*n*25	5	2496	39	1.56	4
*n*33	6	1352	93	6.878	8
*n*40	7	4223	233	5.51	26
*n*50	7	3995	328	8.21	65

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%.

5.2. Comparison of SE_FF, SE_VB, and SE_TS

The proposed SE algorithms have 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 network devices and links were collected from vendors. The characteristics of test cases are listed in Table 3. The smallest test network has 15 local sites and the largest has 50 local sites.

Tables 7–9, respectively, show the best solutions generated by best fixed bias SE_FF, SE_VB, and SE_TS, while Table 10 compares them. It is clear from these tables that, in general, SE_VB produces comparable results with SE_FF as far as "monetary cost" objective is concerned. For "average network delay" and "maximum hops" objectives, a general trend is that SE_FF performs better than SE_VB. As far as execution time is concerned, SE_VB has lower execution time than best fixed bias SE_FF for smaller cases (such as *n*15, *n*25, and *n*33), while for bigger cases (*n*40 and *n*50), SE_VB has higher execution time than SE_FF. However, if we consider the time spent in trial runs of SE_FF algorithm to find the best fixed bias, then SE_VB can be considered better than the fixed bias SE_FF. There were at least three trial runs with different bias values to identify the best value for each test case for SE_FF. For SE_VB, there is no need to run several trials. Figs. 7(a)–(c) show the progression of the two algorithms with respect to the three optimization objectives.

Table 10 also shows the percentage improvement achieved by best tabu list size SE_TS when compared to SE_FF. From these tables, it is seen that SE_TS performs better than SE_FF for monetary cost objective. For all test cases, a gain is achieved by SE_TS. Similarly, for average network delay metric, SE_TS achieves gain in all cases. 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. Fig. 7(d)–(f) show the progression of

Table 7
Best results for SE_FF. B = best bias, C = cost in US $, D = delay in ms/packet, H = hops, T = execution time (min)

Case	SE_FF				
	B	*C*	*D*	*H*	*T*
*n*15	0.2	314,400	3.282	5	4
*n*25	0.2	509,050	4.26	7	5
*n*33	0.0	687,760	4.729	8	40
*n*40	0.3	866,900	4.126	8	12
*n*50	0.3	1061,900	5.32	9	8

Table 8
Best results for SE_VB. C = cost in US$, D = delay in ms/packet, H = hops, T = execution time (min)

Case	SE_VB			
	C	*D*	*H*	*T*
*n*15	305,500	4.135	7	1
*n*25	512,415	4.37	7	4.4
*n*33	702,815	5.319	7	17
*n*40	800,580	6.637	10	42
*n*50	1042,080	8.236	10	62

Table 9
Best results for SE_TS. TL = Tabu list size, C = Cost in US $, D = Delay in ms/packet, H = hops, T = execution time (min)

Case	SE_TS				
	TL	*C*	*D*	*H*	*T*
*n*15	2	297100	2.78	4	2.25
*n*25	5	483210	3.537	6	4
*n*33	6	682465	4.19	6	8
*n*40	7	783970	4.441	9	26
*n*50	7	983020	5.245	11	65

Table 10
Percentage improvement achieved by SE_VB compared to SE_FF and SE_TS compared to SE_FF. C = Cost in US $, D = Delay in ms/packet, H = hops

Case	SE_VB vs. SE_FF			SE_TS vs. SE_FF		
	C	*D*	*H*	*C*	*D*	*H*
n15	2.83	−25.99	−40	5.5	15.29	20
n25	−0.657	−2.51	0	5.07	16.97	14.29
n33	−2.19	−12.48	12.5	0.755	11.4	25
n40	7.65	−60.85	25	9.57	−7.63	−12.5
n50	1.87	−54.8	−11.1	7.43	1.41	−22.2

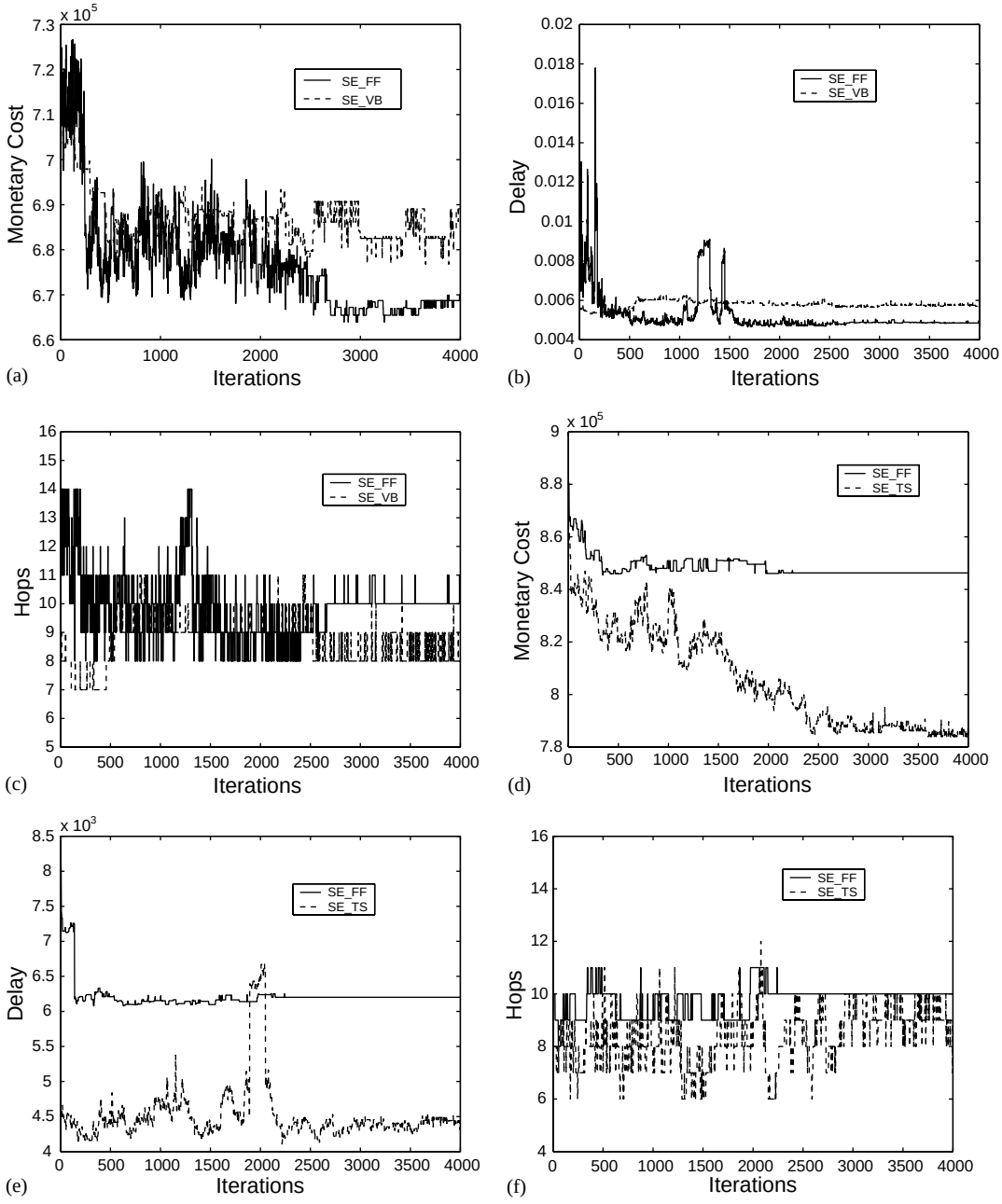

Fig. 7. Progression of objective values with iterations for SE_FF and SE_VB for (a) monetary cost, (b) delay, (c) hops. (d), (e), (f) compare the same objectives, respectively, for SE_FF and SE_TS.

SE_FF and SE_TS for the three optimization objectives. The reason SE_TS has better performance than SE_FF is the following. In SE_FF, since the search space for valid solutions is limited, it happens that after some iterations, same moves are repeated and 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.

6. Conclusion

In this paper, we have presented a novel approach for topology design of enterprise networks based on fuzzy simulated evolution algorithm with three variations. Results obtained for the test cases suggest that fuzzy simulated evolution algorithm with tabu search allocation is a better approach to this problem. Moreover, comparison among the variations showed that the search performed by SE_TS is more intelligent, that is,

the solution subspace investigated by SE_TS is of superior quality than that of SE_FF and SE_VB. Further, as time elapsed, SE_TS progressively evolved toward better solutions, a desirable characteristic of evolutionary heuristics.

Acknowledgements

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

References

Bertsekas, D., Gallager, R., 1992. Data Networks. Second Edition. Prentice-Hall Inc., Englewood Cliffs, NJ.

Black, D.P., 1998. Managing Switched Local Area Networks. Addison-Wesley, Reading, MA.

Dengiz, B., Altiparmak, F., Smith, A., 1997a. Local search genetic algorithm for optimal design of reliable network. IEEE Transactions on Evolutionary Computation 1 (3), 179–188.

Dengiz, B., Altiparmak, F., Smith, A., 1997b. Efficient optimization of all terminal reliable networks, using an evolutionary approach. IEEE Transactions on Reliability 46 (1), 18–26.

Elbaum, R., Sidi, M., 1996. Topological design of local-area networks using genetic algorithm. IEEE/ACM Transactions on Networking 11 (8), 766–778.

Ersoy, C., Panwar, S., 1993. Topological design of interconnected LAN/MAN networks. IEEE Journal on Selected Area in Communications, 11 (8), 1172–1182.

Esau, L.R., Williams, K.C., 1966. On teleprocessing system design. A method for approximating the optimal network. IBM System Journal 5, 142–147.

Fetterolf, P.C., 1990. Design of data networks with spanning tree bridges. in: Proceedings IEEE International Conference on Systems, Man, and Cybernatics 298–300.

Gen, M., Ida, K., Kim, J., 1998. A spanning tree-based genetic algorithm for bicriteria topological network design. in: Proceedings IEEE World Congress on Computational Intelligence, 15–20.

Glover, F., Laguna, M., 1997. Tabu Search. Kluwer Academic Publishers, Dordrecht.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

Kirkpatrick, S., Gelatt Jr., C., Vecchi, M., 1983. Optimization by simulated annealing. Science 498–516.

Khan, S.A., 1999. Topology design of enterprise networks. MS Thesis, King Fahd University of Petroleum and Minerals.

Kling, R.M., Banerjee, P., 1989. ESP: placement by simulated evolution. IEEE Transactions on Computer-Aided Design 8 (3), 245–255.

Pierre, S., Legault, G., 1998. A genetic algorithm for designing distributed computer network topologies. IEEE Transactions on Systems, Man, Cybernatics 28 (2), 249–258.

Prim, R.C., 1957. Shortest connection networks and some generalizations. Bell System Technical Journal 36, 1389–1401.

Sait, S.M., Youssef, H., 1999. Iterative Computer Algorithms and their Application to Engineering. IEEE Computer Science Press, Silver Spring, MD.

Yager, R.Y., 1998. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Transactions on Systems, Man, and Cybernatics 18 (1), 183–190.

Youssef, H., Sait, S.M., Issa, O.A., 1997. Computer-aided design of structured backbones. In: 15th National Computer Conference and Exhibition, pp. 1–18.

Youssef, H., Sait, S.M., Ali, H. 2001. Adaptive bias simulated evolution algorithm for placement. IEEE International Symposium on Circuits and Systems 355–358.

Zadeh, L.A., 1965. Fuzzy Sets. Inform. Control 8, 338–353.