Fuzzy Simulated Evolution Algorithm For Topology Design Of Campus Networks

Habib Youssef Sadiq M. Sait Salman A. Khan

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

Abstract *The topology design of campus networks (CNs) is a hard constrained combinatorial optimization problem. Therefore, we resort to heuristics to come up with desirable solutions. A desirable solution must optimize several conflicting objectives such as minimization of cost, minimization of network delay, and minimization of maximum number of hops etc. Furthermore, some of the objectives are imprecise. Fuzzy Logic provides a suitable mathematical framework in such a situation. In this paper, we present an approach based on Simulated Evolution algorithm for the design of campus network topology. The two main phases of the algorithm, namely, evaluation and allocation, have been fuzzified. We have also incorporated Tabu Search-based characteristics in the allocation phase of the SE algorithm. This approach is then compared with Simulated Annealing algorithm, which is another well-known heuristic. Results show on all test cases that Simulated Evolution algorithm exhibits more intelligent search of the solution subspace and was able to find better solutions than Simulated Annealing.*

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

1 Introduction

A typical campus network consists of a large number of components, such as mainframe computers, mini systems, workstations, PCs, user terminals, printers, etc., [1]. Various devices such as bridges, routers, hubs, multiplexers, etc., are used to interconnect and network these computers and peripherals. The network topology is governed by several constraints. Geographical constraints dictate the break-down of such internetworks into smaller parts or groups of nodes, where each group makes up what is called a LAN. Thus, we can define a campus network as 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, minimizing delay, cabling and equipment cost etc. [1]. 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. Topology design at LAN level requires interconnection of LAN segments via bridges and layer 2 switches [2, 3].

A structured campus network has three layers:

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 packets manipulation takes place.
3. Backbone Layer, which provides high speed optimal transport of data among local sites.

Thus, the design of such a structured campus 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 campus networks has been classified as a hard problem [1]. Even the design of a LAN is itself an NP-hard problem [2, 3, 4]. Therefore, we have to use approximation methods known as 'heuristics' to get near optimal solutions in reasonable amount of time.

In this work, we have used simulated evolution algorithm for topology design of structured campus networks based on several criteria, which are: 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 [5]. According to recommended structured cabling standards, the network topology is constrained to be tree. Hence we target to find a tree topology of desirable quality with respect to the three design objectives. According

Since the backbone design problem is a multi-objective combinatorial optimization problem, 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.

In Section 2, assumptions, terminology, and notation are given. Section 3 describes computation of objective values and constraints. Section 4 presents the proposed scheme. Section 5 gives results and discussion. We conclude in Section 6.

2 Assumptions and Terminology

- A single user (station) is called a "node".
- A "LAN segment" (or simply a segment) is a part of a cluster and is comprised of nodes.
- A "cluster" (also called Local Site) is an interconnected collection of LAN segments.
- 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 (x, y) coordinates with respect to some reference point.
- Each node has a LAN network interface card of a particular technology such as 10/100baseT Ethernet [6] or Token Ring type [6].
- A cluster is constrained to consist of segments only of the same technology.
- The Root node is a switch acting as a collapsed backbone with given required interfaces.
- Between two local sites, only fiber optic cable [7] is used.
- There is a user specified limit on the number of nodes per local site.
- The number of segments is known a priori and nodes have already been assigned to segments.
- Hubs, bridges, routers and other networking devices cannot be placed in any location. They have designated locations.
- Maximum allowed utilization of any link should not exceed a desired threshold (e.g., 60%).

2.1 Notation

n	number of clusters.
m	number of LAN segments in a cluster.
T^b	$n \times n$ local site topology matrix where ${t^b}_{ij} = 1$, if local sites i and j are connected and ${t^b}_{ij} = 0$ otherwise.
ω_i	traffic within cluster i.
λ_i	traffic on link i.
$\lambda_{max,i}$	capacity of link i.
L	number of links of the proposed topology.
D_{nd}	delay due to network devices.
g_i	maximum number of clusters which can be connected to cluster i.
γ_{ij}	external traffic between clusters i and j.
γ	overall external traffic.
S_k	the forwarding speed of network device k.
h_{ij}	number of hops between clusters i and j.

3 Objectives and constraints

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:

$$cost = (l \times c_{cable}) + (c_{nd}) \tag{1}$$

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 [2].

The delay per bit due to network device between local site i and j is $B_{i,j} = \mu b_{i,j}$, where $b_{i,j}$ 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:

$$D_{nd} = \frac{1}{\gamma} \sum_{i=1}^{d} \sum_{j=1}^{d} \gamma_{ij} B_{ij} \tag{2}$$

Thus, total average network delay is composed of delays of links and network devices and is given by [2]

$$D = \frac{1}{\gamma} \sum_{i=1}^{m} \frac{\lambda_i}{\lambda_{max,i} - \lambda_i} + \frac{1}{\gamma} \sum_{i=1}^{n} \sum_{j=1}^{n} \gamma_{ij} B_{ij} \tag{3}$$

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 moves from one network device to another.

3.4 Constraints

Three important constraints are considered.

1. The first set of constraints is due to the limitation of the capacity 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:

$$\lambda_i < \lambda_{max,i} \quad i = 1, 2, ..., e \tag{4}$$

2. The second constraint is that the number of clusters attached to a network device G must not be more than the port capacity of that device.

$$\sum_{j=1}^{n} t_{ij}^{b} < g_i \qquad i = 1, 2, ..., n \qquad \forall i \neq j \tag{5}$$

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

4 Fuzzy Simulated Evolution Algorithm for Network Design

4.1 Simulated Evolution

Simulated Evolution (SE) is a general iterative heuristic proposed in [8]. 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 [9]. This scheme combines iterative improvement and constructive perturbation and saves itself from getting trapped in local minima by using stochastic approach. It iteratively operates a sequence of evaluation, selection and allocation (perturbation) on one solution. The SE proceeds as follows. It starts with a randomly or constructively generated valid initial solution. The main loop of the algorithm consists of three steps: **evaluation**, **selection** and **allocation**. These steps are carried out repetitively in a main loop until some stopping condition is satisfied. Other than these three steps, some input parameters for the algorithm are initialized in an earlier step known as **initialization**.

4.2 Proposed Algorithm and Implementation Details

This section describes our proposals of fuzzification of different stages of the SE algorithm. The description is combined with the implementation details of the SE algorithm for topology design.

Initialization

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

Proposed Fuzzy Evaluation Scheme

The **goodness** of each individual is computed as follows. In our case, an individual is a **link** which interconnects 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 orlike compensatory operator [10], Rule 1 translates to the following equation for the fuzzy goodness measure of a link l_i.

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

The superscript e stands for **evaluation** and is used to distinguish similar notation in other fuzzy rules. In equation 6, $\mu^e(l_i)$ 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 following manner. From the cost matrix, which gives the costs of each possible link, we find the minimum cost of all the costs and maximum cost of all the costs. We take these minimum and maximum costs as the lower and upper bounds and call them "LCostMin" and "LCostMax" respectively and then find the membership of a link with respect to these bounds. Furthermore, In this work, we have normalized the monetary cost with respect to "LCostMax". The required membership function is represented as depicted in Figure 1:

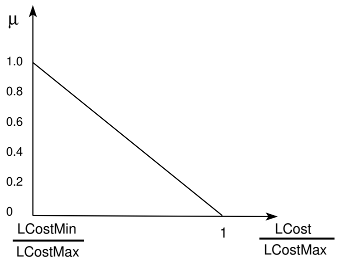


Figure 1: Membership function for monetary cost of a link.

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. This is logical to take as the lower bound since there can be no link which is connected to the root at a depth of zero (see Figure 2). 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 7. 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 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 protocol 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 Figure 3:

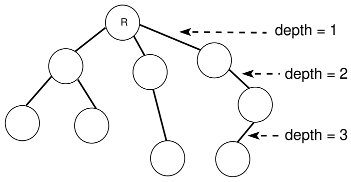


Figure 2: Depths of links with respect to the root.

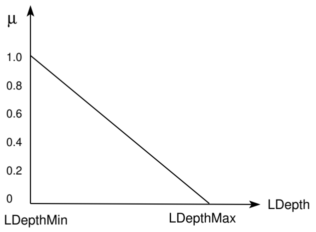


Figure 3: Membership function for depth of a link.

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* [11] has been used in this paper. The *variable bias* is a function of *quality of solution*. When the overall solution quality is bad, 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:

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

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

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, 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, as depicted in Figure 4.

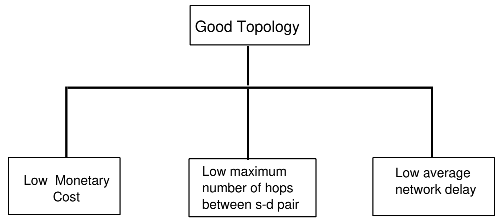


Figure 4: Basic components for a good topology.

The following rule and the subsequent equation are used for this purpose.

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*.

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

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, ${\mu_i}^a$ 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 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 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, 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 initialization step. The monetary cost is normalized with respect to "TCostMax". The corresponding membership function is shown in Figure 5.

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 3. The maximum value of average delay,"TDelayMax", is taken to be the average delay generated in the initialization step. The average delay is normalized with respect to "TDelayMax". The membership function is shown in Figure 6.

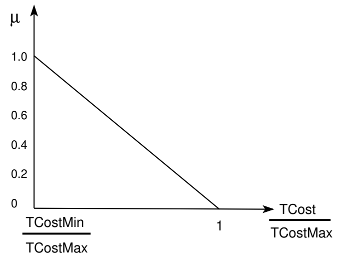


Figure 5: Membership function for monetary cost.

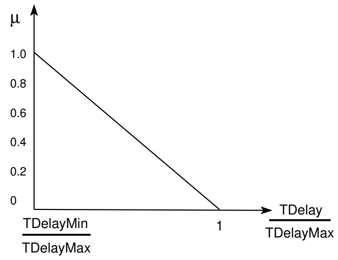


Figure 6: Membership function for average network delay.

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 case. The maximum value,"THopsMax", is taken to be the maximum number of hops between any source-destination pair generated in the initialization step. The membership function is shown in Figure 7.

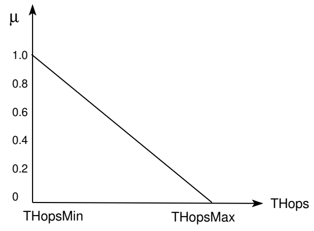


Figure 7: Membership function for maximum number of hops between a source-destination pair.

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 8.

Now the placing of trial links begins. In this work, the approach to place trial links is as follows. At most ten trial moves (i.e., trial links) are evaluated for each removed link. One point to mention is that for the ten moves, some moves may be invalid. However, we search for only four "valid" moves. Whenever we find four valid moves, we stop, otherwise we continue until a total of ten moves are evaluated (whether valid or invalid). The removal of a link involves two nodes P and Q, of which node P belongs to the subtree which contains the root node and node Q belongs to the other subtree, as shown in Figure 8. For the ten moves we make, five of them are controlled and five are random. For the controlled 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.

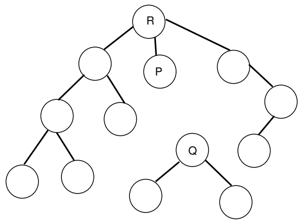


Figure 8: Two disjoint graphs containing nodes P and Q.

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 7 and the best move among the ten moves is made permanent. This procedure is repeated for all the links that are present in the set of selected links. 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* is that if a 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. For details on tabu search, refer to [12].

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

The SE algorithm described in this paper, namely SE, 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 links were gathered 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.

We compare the proposed SE with SA. Recall that 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 α=0.9, β=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, a gain is 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 in performance 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 9(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, then the original link is placed back to 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 9(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.

Figures 9(b) and (c) illustrate very interesting and desirable behavior of SE. The figures plot the cumulative frequency of solutions for different membership ranges after each 200 iterations for **n25**. In Figure 9(b), the plot for SA shows that most of the solutions are falling in 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 9(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 memberhsip ranges remains the same, suggesting that with time, the algortihm is evolving towards better solution subspaces.

6 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.

Acknowledgments

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

References

[1] Habib Youssef, Sadiq M. Sait, and Osama A.Issa. Computer-Aided Design of Structured Backbones. In *15th National Computer Conference and Exhibition*, pages 1–18, October 1997.

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

[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*, pages 164–173, May 1998.

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

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

[6] Mischa Shwartz. *Telecommuncations Networks: Protocols, Modeling, and Analysis.* Addison-Wesley Publishing Company, 1987.

[7] Gerd. E. Keiser. *Local Area Networks.* McGraw-Hill Book Compnay., 1989.

[8] Ralph Michael Kling. *Optimization by Simulated Evolution and its Application to cell placement.* Ph.D. Thesis, University of Illinois, Urbana, 1990.

[9] D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, Jan 1994.

[10] Ronald Y. Yager. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking. *IEEE Transactions on Systems, Man, and Cybernatics*, 18(1):183–190, Jan 1988.

[11] Ali S. Hussain. *Fuzzy Simulated Evolution Algorithm for VLSI Cell Placement.* MS Thesis, King fahd University of Petroleum and Minerals, 1998.

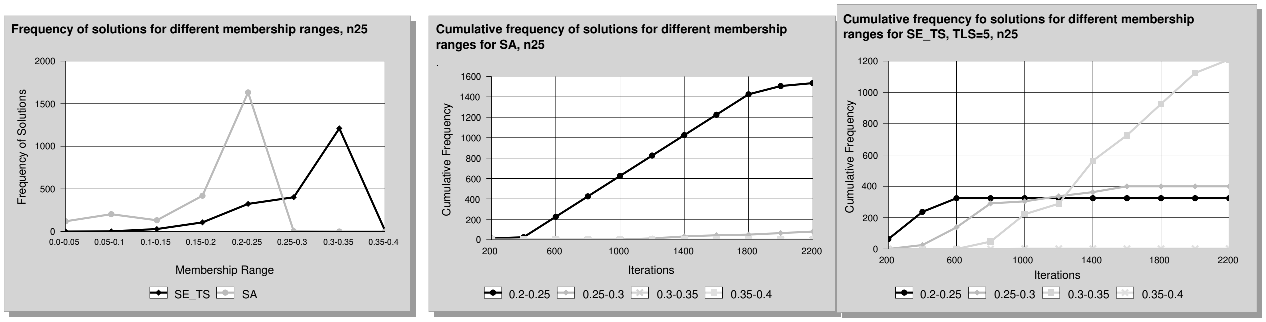
[12] Sadiq M. Sait and Habib. Youssef. *Iterative Computer Algorithms and their Application to Engineering.* IEEE Computer Science Press, Dec. 1999.

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

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

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

Table 2: Comparison of SA and SE. C = Cost in US \$, D = Delay in milli seconds 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.



(a) (b) (c)

Figure 9: Comparison of SE algorithm with SA for different membership ranges in fuzzy set "good topology". (a), (b) & (c) respectively compare the frequency of solutions, cumulative frequency of solutions for SA, and cumulative frequency of solutions for SE