Available online at www.sciencedirect.com

ScienceDirect

Computer Communications 30 (2007) 2010–2020

computer communications

www.elsevier.com/locate/comcom

Applications of genetic algorithms to optimal multilevel design of MPLS-based networks

El-Sayed M. El-Alfy *

College of Computer Sciences and Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

Received 24 May 2006; received in revised form 4 March 2007; accepted 8 March 2007
Available online 13 March 2007

Abstract

This paper proposes a design methodology based on the application of genetic algorithms (GA) to find a minimal-cost topological structure of MPLS-based networks. MPLS technology is currently deployed in designing the backbone infrastructure of service provider networks whereas other parts of the network are still operated using the traditional IP protocol. This makes the overall topological structure of MPLS-based networks naturally breaks into two prime sub-problems: access network design and backbone network design. The ultimate goal is to identify the locations of label-edge routers and label-switching routers, and to determine the interconnection links and their capacities to accommodate expected traffic demands. The locations of label edge routers depend on the demands of a given set of terminal networks which in turn affect the design of the backbone network. This problem is a highly constrained NP-hard optimization problem for which exact solution approaches do not scale well. We first present a multilevel design model that divides the optimal topology design into a set of linear programs. Then, we propose GA-based meta-heuristics for solving them. We also discuss the impact of encoding methods and genetic operators and parameters on the performance. Numerical results for the considered cases show that the proposed methodology is effective and gives optimal or close to optimal solutions as compared with the exact branch and bound method.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Genetic algorithms; Network optimization; Multiprotocol-label switching

1. Introduction

Multiprotocol-label switching (MPLS) [1,2] provides a promising approach for supporting differentiated quality-of-service (QoS) required for multimedia delivery over the Internet. It incorporates a wide-range of capabilities that combine the merits of both circuit-switched and packet-switched networks. One of the current applications of MPLS technology is in the backbone of service provider networks. Building a cost-effective network that meets its business and technical goals is a daunting endeavor. This problem is a highly constrained optimization problem for which exact solution approaches do not scale well. Over years, network researchers and practitioners have developed several models and heuristic algorithms to reduce the computational intricacy of the problem. The goal is to identify geographical locations of nodes (concentrators/multiplexers, switches, routers, etc.), network connectivity and link capacity to accommodate expected traffic demands with reasonable cost-performance tradeoff. Unlike other published work on topology design, MPLS has two distinct sets of nodes: label-edge routers (access nodes) and label-switching routers (transit nodes) which complicate the problem further. In this paper, we focus on hierarchical network design in which the backbone infrastructure is implemented using MPLS technology and low level networks utilize the traditional IP protocol. We also assume that the backbone and the low level networks are owned and operated by independent institutions. Under this assumption, the design of the entire network can be divided into two independent sub-problems. First, an access network is designed to accommodate the traffic demands of a given set of terminal networks. Second, a

* Tel.: +966 3 860 1930; fax: +966 3 860 2174.
E-mail address: alfy@kfupm.edu.sa.

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.03.005

service provider builds a cost-effective infrastructure to connect the selected access nodes. While traffic demands in low level networks are typically centralized, the backbone network traffic demand is generally distributed and hence the backbone topology is arbitrarily chosen [3]. After describing the formal representation of these sub-problems, we present solution approaches based on genetic algorithms. We also discuss the impact of encoding methods and genetic operators and parameters on the performance. The effectiveness of the proposed methodology is evaluated for a number of examples and the results are benchmarked to optimal solutions obtained using the exact branch and bound method. As mentioned in [4], planning hierarchically is easier than optimizing the network as a whole since different parts of the hierarchy can be treated independently by different staff members who maintain and modify the network. In addition, a minimum cost for the entire network can still be reached by applying the design at different levels iteratively [3].

The rest of this paper is organized as follows. In the next section, we first survey related work on network topology design. Then, we review some architectural aspects of MPLS networks in Section 3 and formulate the design problem in Section 4. The proposed solution based on genetic algorithms is presented in Section 5. In Section 6, we provide simulation results and compare the attained solutions with the exact optimum. Finally, Section 7 concludes the paper and highlights some possible future research directions.

2. Related work

Network planning and optimal topological design has been an area of extensive research since the early days of computer networks [5]. Several studies have been published on topology design for circuit-switched and packet-switched networks [6–8]. In general, the network design problem is formulated as an optimization problem with the objective of minimizing (or maximizing) a cost function (or a performance metric) subject to a set of constraints. Different performance metrics can be used such as average packet delay, average hop count, link utilization, etc. Constraints can vary based on geographical constraints, technology constraints, performance constraints, etc. A number of mathematical programming techniques such as linear programming, integer programming and mixed-integer programming are commonly used for solving it.

Dealing with such problem in today's large-scale networks is a complex endeavor. Since the complexity of this problem is known to be NP-complete, several heuristic algorithms based on simulated annealing, evolutionary strategies and genetic algorithms have been proposed in the literature to a number of related optimization problems [9–11]. Although simple genetic algorithms were introduced for unconstrained numerical function optimization, several approaches have been proposed for extending them to handle constraints [12]. The terminal assignment problem, for clustering terminals and connecting them to concentrators, is addressed in [13] using heuristic approaches based on greedy algorithms and genetic algorithms. In [14], a heuristic approach based on genetic algorithms is described for solving the topology design of local-area networks with the objective of minimizing the average network delay. Designing a campus network topology is addressed in [15] using an evolutionary algorithm based on fuzzy simulated evolution with Tabu search. In [16], an evolutionary algorithm has been applied to telecommunication network dimensioning to find values of link capacities. One of the best-known network optimization problems that is used for designing backbone networks is minimum-spanning tree (MST) problem. In [17], GAs have been used for solving a variation of MST called degree-constrained MST. The study in [18] has demonstrated the topology design of B-ISDN networks using genetic algorithms. The same problem is again addressed in the communications letter [19] using steady state genetic algorithms. Unlike the work we consider in this paper, the authors assumed uncapacitated problem with given node locations and all nodes are of the same type. A commonly used heuristic approach for solving the node location problem is known as Add-Heuristic [6]. The topology design of MPLS core network was considered in [20] and formulated as a mixed-integer program. Since this problem is known to be NP-complete for which there is no algorithm known to run in polynomial time, the authors proposed a heuristic approach based on branch-and-bound (BB) algorithm for solving it. In [21], the author proposed greedy randomized adaptive search procedures (GRASP) for solving the topological design problem of MPLS networks. In this study, we investigated another heuristic approach based on genetic algorithms for solving it; preliminary results have been partially published in [22].

3. MPLS network architecture

MPLS is built on former ideas of tag switching and label switching to expedite packet forwarding and traffic engineering in packet-switched networks such as the Internet [1]. A typical MPLS network architecture is illustrated in Fig. 1. The MPLS network topology can be divided into MPLS core and MPLS edge. A router supporting MPLS at the edge is termed as a Label Edge Router (LER) where incoming packets are classified into Forwarding Equivalence Classes (FECs). This classification is based on the network-layer information contained in the packet and any other control information available at the router such as balancing supported QoS and network utilization. Packets in the same FEC are forwarded over the same path and are treated in the same manner. A path from an ingress LER router to an egress LER router is termed as Label Switching Path (LSP) or tunnel. Intermediate routers are MPLS-capable routers and are called Label Switching Routers (LSRs).

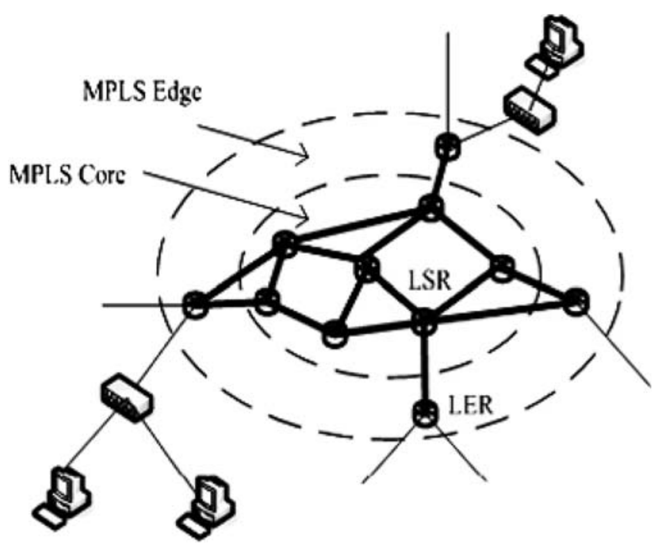


Fig. 1. A typical MPLS network architecture.

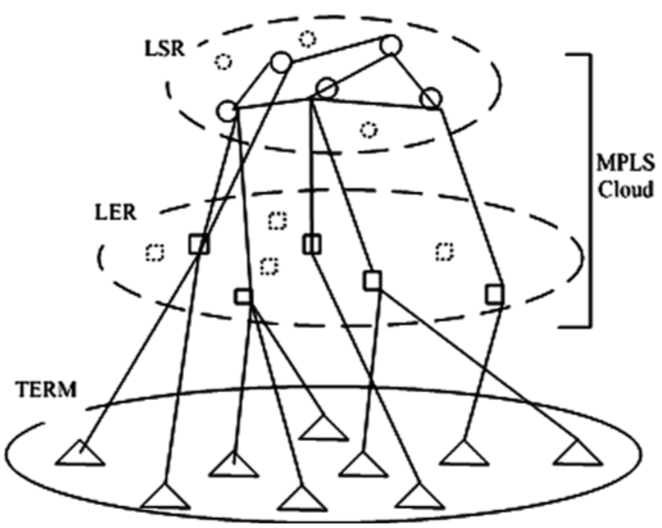


Fig. 2. MPLS layered topology design.

LSR nodes are similar to LER nodes but they are either not capable of analyzing the network-layer headers at all, or not capable of doing that at adequate speed. LER nodes are the connection points between non-MPLS and MPLS-capable routers hence they are also called access nodes. LSR nodes are capable of doing label lookup and replacement and are called transit nodes. The label distribution protocol (LDP) or the resource reservation protocol (RSVP) can be used to set up and manage tunnels.

4. Methodology and problem formulation

In this section, we describe a hierarchical design methodology and present a mathematical programming formulation for the optimal topology design of MPLS-based networks. For given expected traffic demands, the objective is to determine the optimal location of nodes, and the interconnection links and their capacities to minimize the overall costs while satisfying a number of business and technical constraints. To simplify the design problem, we use a two-layer approach, as illustrated in Fig. 2, to handle these interrelated questions separately and combine the solutions to answer the overall design problem. We start by clustering terminals or access networks and assigning them to a subset of a given set of candidate locations for access nodes (LERs). Then, we consider the problem of identifying locations of LSR nodes and determining the interconnection links and their capacities between LSR nodes and between LER nodes and LSR nodes.

4.1. Terminal assignment (TA)

Before looking at the general problem of identifying access node locations and assigning terminals to them, we first address the simplified assignment problem where access nodes are given and are assumed to be fixed. The objective now is to find a best assignment of access networks (terminals) to access nodes (LERs) in order to minimize the connection costs. Although the associated link cost can be based on distance, delay, capacity, etc., in the numerical examples we use Euclidean distance. Let $I = \{1, 2, \ldots, N\}$ be the set of terminals and $J = \{1, 2, \ldots, M\}$ be the set of access nodes. We define $x_{ij} = 1$ if terminal $i \in I$ is connected to access node $j \in J$ and $x_{ij} = 0$ otherwise. The cost of connecting terminal $i \in I$ to access node $j \in J$ is denoted by c_{ij}. A terminal must be assigned to only one access node and each access node can handle up to a certain number of terminals as given by the vector $\boldsymbol{k} = (k_1, k_2 \ldots, k_M)$. This problem is known as terminal assignment (TA) problem and is usually formulated as a combinatorial optimization problem. Table 1 summarizes the TA formulation as a 0/1 integer program.

Terminals may have different capacity requirements and in such case the capacity constraints will be modified as follows:

$$\sum_{i=1}^{N} b_i x_{ij} \leqslant k_j \quad \forall j \in J,$$

where b_i is the bandwidth required by terminal i and k_j is the maximum bandwidth of node j. Additional constraints may be imposed such as type of service available at each node or link.

Table 1
Integer program formulation for terminal assignment

$\min F(\boldsymbol{x}) = \sum_{j=1}^{M}\sum_{i=1}^{N} c_{ij}x_{ij}$	(1a)
subject to	
$\sum_{j=1}^{M} x_{ij} = 1, \forall i \in I$	(1b)
$\sum_{i=1}^{N} x_{ij} \leqslant k_j, \forall j \in J$	(1c)
$x_{ij} \in \{0,1\},\ \forall i \in I;\ \forall j \in J$	(1d)

4.2. Access node location (ANL)

This problem is similar to TA except that the locations of LER nodes are not known but only a candidate set of possible locations, J, is given. The goal is to select a subset of J at which LER nodes are installed and to determine links connecting terminals to them in order to minimize the overall installation costs. In addition to link installation costs, there is a node installation cost for each selected location. The cost information is given. Again, this problem can be formulated as a combinatorial optimization problem as summarized in Table 2. We introduced a new set of decision variables, $y_j \in \{0,1\}$, such that $y_j = 1$ if we decided to install an LER at location $j \in J$; otherwise $y_j = 0$. Also the objective function is modified to include nodal costs as follows:

$$\min\left\{\sum_{j=1}^{M}\eta_j y_j + \sum_{j=1}^{M}\sum_{i=1}^{N} c_{ij}x_{ij}\right\},$$

where η_j is the installation cost for an LER at location $j \in J$.

Similarly, we can assume terminals have different bandwidth requirements and in such case the capacity constraints will be modified to be,

$$\sum_{i=1}^{N} b_i x_{ij} \leqslant k_j y_j \quad \forall j \in J.$$

4.3. Transit node location and connectivity (TNLC)

After the locations of LER nodes have been identified and connecting access networks to them, it is required to provide connectivity between LER nodes to accommodate the expected traffic demand. Hence, in this section we assume that the number and locations of LER nodes are given. There is a candidate set of possible transit node locations. The expected traffic demand between each pair of source–destination LER nodes and the admissible paths are also given. Without loss of generality, we assume demand is bidirectional. Installing a transit node at a specific location incurs a fixed installation cost. Connecting two nodes has a fixed link installation cost and a varying cost depending on the link load. Our goal here is to determine (1) the optimal locations of LSR nodes, (2) the interconnections between all nodes, (3) flow pattern for realizing each demand. An optimal solution to this problem is a minimum-cost topology that accommodates demands while satisfying a number of specified constraints such as maximum link utilization, link capacity, number of interfaces at each node, different levels of service quality in terms of delay, bandwidth, hop count, etc. The mathematical formulation of this topology design problem is stated in Table 3 as a mixed-integer program.

The input data to the optimization procedure consists of:

- A set of LER nodes $I = \{1,2,\ldots,N\}$ described by locations $(x_1,y_1),\ldots,(x_N,y_N)$.
- A set of demands, D, described by the vector $\boldsymbol{h} = (h_d\text{: } d \in D)$ that specifies the demand volume between each source–destination pair of LER nodes.
- A set of candidate locations $J = \{1,2,\ldots,M\}$ of LSR nodes described by coordinates $(x_1,\ y_1),\ldots,(x_M,y_M)$.
- A set of all possible links, E.
- Admissible paths for realizing the demand between each pair of access nodes (defined by link-path indicator matrix $\boldsymbol{u} = (u_{edp}\text{: } e \in E,\ d \in D,\ p \in P_d)$ of order $|E| \times |D| \cdot |P_d|$ where $u_{edp} = 1$ if path p for demand d is passing through link e; otherwise $u_{edp} = 0$. P_d is the set of paths for realizing demand d for all $d \in D$.
- Transit node incidence matrix $\boldsymbol{\phi} = (\phi_{ev}\text{: } \forall e \in E, \forall v \in J)$ of order $|E| \times |J|$ where $\phi_{\mathrm{ev}} = 1$ if link $e \in E$ is connected to the transit node $v \in J$.
- Link capacity/bandwidth vector $\boldsymbol{b} = (b_e\text{: } \forall e \in E)$ represents upper bound on link capacity if it exists; otherwise the capacity is zero.
- Transit node maximum degree $\boldsymbol{k} = (k_v\text{: } \forall v \in J)$.
- Transit node location cost vector $\boldsymbol{\eta} = (\eta_v\text{: } \forall v \in J)$ represents fixed installation costs.
- Link fixed installation fee and capacity-dependent costs: $\boldsymbol{f} = (f_e\text{: } \forall e \in E)$ and $\boldsymbol{c} = (c_e\text{: } \forall e \in E)$ where $\boldsymbol{c}$ is defined per capacity unit. The output of the procedure is a minimum-cost network defined by a tuple $(\boldsymbol{n}, \boldsymbol{\omega}, \boldsymbol{x}, \boldsymbol{y})$ where
- $\boldsymbol{n}$ is a vector of binary values such that $n_v = 1$ if a transit node is installed at location $v \in J$; otherwise, $n_v = 0$.
- $\boldsymbol{\omega}$ is a vector of binary values such that $\omega_e = 1$ if link $e \in E$ is installed; otherwise, $\omega_e = 0$.
- $\boldsymbol{x} = (x_{dp}\text{: } \forall d \in D,\ p \in P_d)$ is a non-negative real-valued flow pattern for realizing demands over admissible paths such that

$$\sum_{p} x_{dp} = h_d, x_{dp} \in \Re,\ \ \forall d \in D.$$

Table 2
Integer program formulation for access node location (ANL)

$\min F(\boldsymbol{x},\boldsymbol{y}) = \sum_{j=1}^{M}\eta_j y_j + \sum_{j=1}^{M}\sum_{i=1}^{N} c_{ij}x_{ij}$	(2a)
subject to	
$\sum_{j=1}^{M} x_{ij} = 1, \forall i \in I$	(2b)
$\sum_{i=1}^{N} x_{ij} \leqslant k_j y_j, \forall j \in J$	(2c)
$x_{ij} \in \{0,1\},\ \forall i \in I;\ \forall j \in J$	(2d)
$y_j \in \{0,1\},\ \forall j \in J$	(2e)

Table 3
Mixed integer program formulation for transit node location and connectivity (TNLC)

$\min F(\boldsymbol{n},\boldsymbol{\omega},\boldsymbol{x},\boldsymbol{y}) = \sum_v \eta_v n_v + \sum_e [f_{ie}\omega_e + c_e y_e]$	(3a)
subject to	
$\sum_p x_{dp} = h_d,\ \forall d \in D$	(3b)
$\sum_d \sum_p u_{edp} x_{dp} = y_e,\ \forall e \in E$	(3c)
$\sum_e \phi_{ev}\omega_e \leqslant k_v n_v,\ \forall v \in J$	(3d)
$y_e \leqslant b_e \omega_e,\ \forall e \in E$	(3e)
$\omega_e \in \{0,1\},\ \forall e \in E$	(3f)
$n_v \in \{0,1\},\ \forall v \in J$	(3g)

- $\boldsymbol{y} = (y_e\colon \forall e \in E,\ y_e \in \Re)$ is a non-negative real-valued vector representing link loads.

5. GA-based solution

This section explores the application of genetic algorithms for solving the optimization problems formulated in the previous section. Genetic Algorithms are one of the most powerful and broadly applicable guided stochastic search techniques for global optimization. These procedures are based on the principles and mechanisms of natural selection and genetic sciences of biological organisms. A genetic algorithm starts off with a randomly generated population of individuals (also called chromosomes). Each individual represents a solution to the optimization problem and is associated with a fitness value. New solutions are formed by recombining and perturbing existing solutions in the current population based on their fitness to survive. The algorithm repeatedly applies selection, crossover, mutation and replacement operators until a satisfactory solution is found or a maximum number of iterations is reached. Over time, the fittest individuals will have a better chance of surviving and the least fit individuals will be eliminated. Unlike other optimization techniques, genetic algorithms make few assumptions about the problem domain and thus can be applied to a wide spectrum of problems.

The link between genetic algorithms and the optimization problem lies in the encoding procedure to represent decision variables (phenotypes) into chromosomes (genotypes) and computing fitness values. Fitness values are non-negative values derived from the objective function values. Constraints can be handled in different ways [12]. A direct approach is to devise a representation and genetic operators that ensure feasibility. But this is not always obvious and makes the solution problem dependent. Alternatively, we can drop infeasible solutions or use a repair function to transform an infeasible solution into a feasible one. Finally, we can allow infeasible solutions to be in the population but penalize them by adding a penalty term to the objective function. After termination the fittest chromosome is decoded back into the corresponding phenotype. In addition to encoding and fitness evaluation that are problem dependent, the design of a GA-based solution involves other issues including selection, crossover, mutation and replacement strategies.

Selection. Selecting parent chromosomes for mating can be done in various ways. We use a common approach called *weighted roulette-wheel selection* where members are selected randomly but proportional to their relative fitness values. This gives credit for good members and balance required exploration of new regions in the solution space. In the following, we assume a cost minimization problem. Hence, a good chromosome is one that has low relative fitness.

Crossover. Two selected parents from the population are combined to produce two new individuals (offspring) by partially exchanging their genes. For example, in single-point crossover, offspring are formed by swapping bits after the crossover point between the two parents. The motivation is that two good parents are more likely to produce better children even than themselves. The crossover is controlled by the crossover probability, p_c, which is typically in the range [0.7–.95].

Mutation. One or more genes of a chromosome are changed randomly to allow other solutions to be explored. This perturbation improves the performance of GA and prevents a premature convergence to a local minimum. The mutation rate greatly affects the performance of the algorithm. Too much mutation badly affects the results. Typical values for the mutation probability, p_m, are in the range [0.01–0.2].

Replacement. A new population is formed by replacing individuals in the current population with the newly generated offspring. A single scalar called generation gap, $ggap \in [0, 1]$, is used to control the number of replaced individuals. In one implementation, a new generation of size equal to the population size entirely replaces the current population ($ggap = 1$). This is known as generational genetic algorithms (GGA). A variation of this approach is to allow the best individuals to propagate from the current population to the new population. This is known as GGA with elitism. If only one or two individuals are replaced at each iteration, GA is said to be incremental or steady state (SSGA).

In the following, we show how to encode and evaluate individuals in the population for each part of the topology design problem.

5.1. GA-based terminal assignment (GATA)

Encoding method. The first step in designing a genetic algorithm solution is to devise a suitable encoding scheme. A solution to the TA problem is represented by an integer vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ where $x_i = j$ is the jth LER node to which the ith terminal is assigned. This encoding method guarantees that each terminal is assigned to only one LER node. For a solution vector, $\boldsymbol{x}$, terminals assigned to the jth LER are determined by the set $\{i\colon x_i = j\}$. A feasible solution is a vector, $\boldsymbol{x}$, such that the number of terminals assigned to each LER is not exceeding its capacity (node degree), i.e., $|\{i\colon x_i = j\}| \leqslant k_j,\ \forall j \in J$. This problem can be solved using binary string representation as well which can be done in different ways. For example, each integer can be represented using a fixed number of bits equal to $\lceil \log_2 M \rceil$. The chromosome length is thus $N \lceil \log_2 M \rceil$. In such case, many infeasible solutions are generated because a terminal can be connected to more than one LER which is not allowed. These infeasible chromosomes need to be penalized to reduce their chance in participation in the evolution process. In another representation a chromosome can be of NM bits. In this case, each M bits should have only one bit equal to 1 indicating the LER to which the terminal is connected. But again this binary representation does not ensure feasibility.

Fitness evaluation. Each chromosome is assigned a fitness value derived from the objective cost function. If $\boldsymbol{x}$ is a feasible solution, then

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} c_{ix_i}$$

If infeasible solutions can exist in the population, we use a penalty function to handicap them. The fitness value is defined using a modified objective function as given by

$$\tilde{f}(\boldsymbol{x}) = f(\boldsymbol{x}) + p(\boldsymbol{x}),$$

where $p(\boldsymbol{x})$ defines the penalty value in terms of node overloads and/or number of unsatisfied constraints. If we define the load of node j at solution $\boldsymbol{x}$ to be $g_j(\boldsymbol{x})$ then the node overload is $\max(0, g_j(\boldsymbol{x}) - k_j)$ and the number of unsatisfied constraints is $|\{j\colon \max(0, g_j(\boldsymbol{x}) - k_j) > 0\}|$. In our experimental work, we defined $p(\boldsymbol{x})$ using one of the following formulas:

$$p_1(\boldsymbol{x}) = \delta|\{j : \max(0, g_j(\boldsymbol{x}) - k_j)\}|,$$

$$p_2(\boldsymbol{x}) = \delta\sum_{j=1}^{M} \max(0, g_j(\boldsymbol{x}) - k_j),$$

$$p_3(\boldsymbol{x}) = \delta\sum_{j=1}^{M} \max(0, g_j(\boldsymbol{x}) - k_j) \times |\{j : \max(0, g_j(\boldsymbol{x}) - k_j)\}|,$$

$$p_4(\boldsymbol{x}) = \left[\sum_{j=1}^{M} \max(0, g_j(\boldsymbol{x}) - k_j) + \delta\right] \times |\{j : \max(0, g_j(\boldsymbol{x}) - k_j)\}|.$$

The penalty factor δ is set to a large value so that each infeasible solution becomes worse than the feasible ones. When all constraints are satisfied (i.e., a feasible solution), the penalty will be zero and the fitness value will be the same as the objective function value of the corresponding phenotype.

5.2. GA-based access node location (GANL)

A solution to this problem is represented similarly as in GATA using integer vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$. The decision vector, $\boldsymbol{y}$, is implicitly encoded. When evaluating a chromosome, $\boldsymbol{x}$, we first find $\boldsymbol{y}$ then if the node load is zero, it is not installed and its installation cost is not added to the objective function value. Otherwise its installation cost is added to the objective function value. In our experimental work, whenever we allow infeasible solutions to be in the population, we use penalty functions similar to those used in GATA. After the program terminates, we determine the vector $\boldsymbol{y}$ from the best chromosome.

5.3. GA-based transit node location and connectivity (GATNLC)

As mentioned in Section 4.3, a solution is defined by a tuple $(\boldsymbol{n}, \boldsymbol{\omega}, \boldsymbol{x}, \boldsymbol{y})$. This solution will be represented as a binary string corresponding to the flow pattern, $\boldsymbol{x}$,

$$\boldsymbol{x} = (\boldsymbol{x}_d : d = 1, \ldots, D); \;\; \boldsymbol{x}_d = (\boldsymbol{x}_{dp} : p = 1, \ldots, P_d).$$

The link load vector $\boldsymbol{y}$, and the node and link status vectors ($\boldsymbol{n}$ and $\boldsymbol{\omega}$) are implicitly encoded in the flow pattern since we can compute them from $\boldsymbol{x}$ as follows:

- The link load is obtained by adding all flows passing through the link, $\boldsymbol{y} = \boldsymbol{x} \cdot \boldsymbol{u}^{\mathrm{T}}$, where $\boldsymbol{u}^{\mathrm{T}}$ is the transpose of matrix $\boldsymbol{u}$,
- If the carried load on the link is not zero, then the link should exist, i.e., $\boldsymbol{\omega} = (\boldsymbol{y} \neq 0)$,
- If a link is provided then its end nodes are, $\boldsymbol{n} = (\boldsymbol{\omega} \cdot \boldsymbol{\phi}^{\mathrm{T}} \neq 0)$ where $\boldsymbol{\phi}^{\mathrm{T}}$ is the transpose of matrix $\boldsymbol{\phi}$.

The fitness value is defined using the objective function value if only feasible solutions are allowed in the population. Otherwise, we use a penalty function in which the penalty factor increases over generations, t, as follows:

$$p(\boldsymbol{x}) = (Ct)^{\alpha} \sum_{i=1}^{m} G_i(\boldsymbol{x}),$$

where

$$G_i(\boldsymbol{x}) = \begin{cases} [\max(0, g_i(\boldsymbol{x})]^{\beta} & \text{for } i = 1, \ldots, q \\ |g_i(\boldsymbol{x})|^{\gamma} & \text{for } i = q+1, \ldots, m \end{cases}$$

m is the number of constraints,q is the number of inequality constraints,$m - q$ is the number of equality constraints,t is generation count,C, α, β, γ are penalty parameters,$g_i(\boldsymbol{x})$ is the difference between the left- and right-hand sides of constraint i.

6. Numerical examples

In this section, we present a number of numerical examples to test the effectiveness of the proposed approach. The proposed solutions were implemented and simulations were carried out using MATLAB and the genetic algorithms toolbox [23].

6.1. Terminal assignment

We tested the proposed method GATA for assigning terminals to LER nodes by using 25 terminal networks and 5 LER nodes with capacities defined by the vector $\boldsymbol{k} = (8, 6, 4, 3, 4)$. The locations of terminals and LER nodes are randomly generated on a square layout of size 200×200 as depicted in Fig. 3(a). The assignment cost is taken as the length of the link connecting a terminal to an LER node (Euclidian cost). An optimal solution is found using the branch-and-bound (BB) method which is

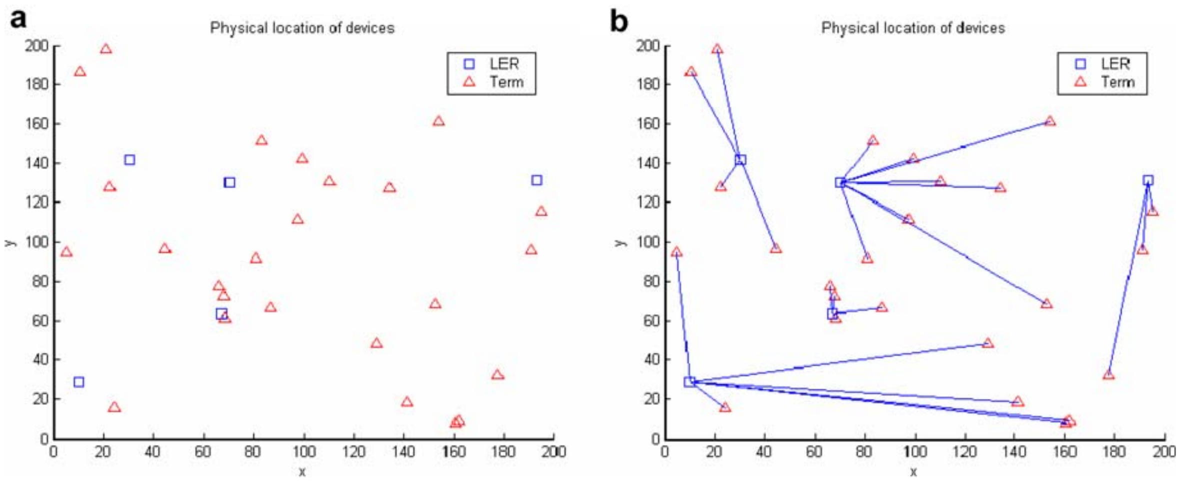


Fig. 3. (a) Randomly generated terminal and LER locations. (b) An optimal assignment found using BB method.

implemented in MATLAB for solving binary integer programming problem as shown in Fig. 3(b) for which the objective function value is 1436.2373. It is clear that each terminal is assigned to only one LER and the number of terminals connected to an LER does not exceed its capacity (node degree). Then we applied GATA to the same example with the following settings: population size $p_s = 100$, single-point crossover with crossover rate $p_c = 0.8$, mutation rate $p_m = 0.01$, and fitness-based replacement with generation gap $ggap = 0.02$. The initial population starts off with a set of feasible solutions and during reproduction infeasible offspring are dropped. The corresponding changes of the best and average fitness values versus generations are shown in Fig. 4(a) and the best assignment found is illustrated in Fig. 4(b). We can see that the algorithm converges quickly to the best solution for these parameter settings. After 10,000 generations, the objective function value of the best solution is 1436.2371 which is exactly equal to the optimal solution found using the BB methods.

In the following experiments, we allowed infeasible solutions to exist in the initial population and during reproduction but penalize them. After generating an initial population uniformly random, we evaluate each chromosome and penalize it if it represents an infeasible solution. Also during reproduction, infeasible offspring are penalized and inserted in the new generation. As mentioned before the penalty function makes the fitness value worse and reduces the likelihood of selecting these chromosomes for mating. We first defined the penalty function as $p_4(\boldsymbol{x})$ defined in Section 5.1 with $\delta = 400$. The best and average fitness variations and the best assignments are shown in Fig. 5 for which the objective function value is 1465.4168. Comparing it with the optimal solution, it is only 2% higher than the optimum. We can also see that the convergence becomes slower than if we drop infeasible solutions. We tried to use other penalty functions such as $p_3(\boldsymbol{x})$ with $\delta = 1{,}000$. From the results shown in Fig. 6, we can that it gives better solution (less than 0.04% higher than the optimum). We carried out several other experiments with different parameter settings and penalty functions [24]. It is found in some implementations of the GA-based solution that the results are either optimal or

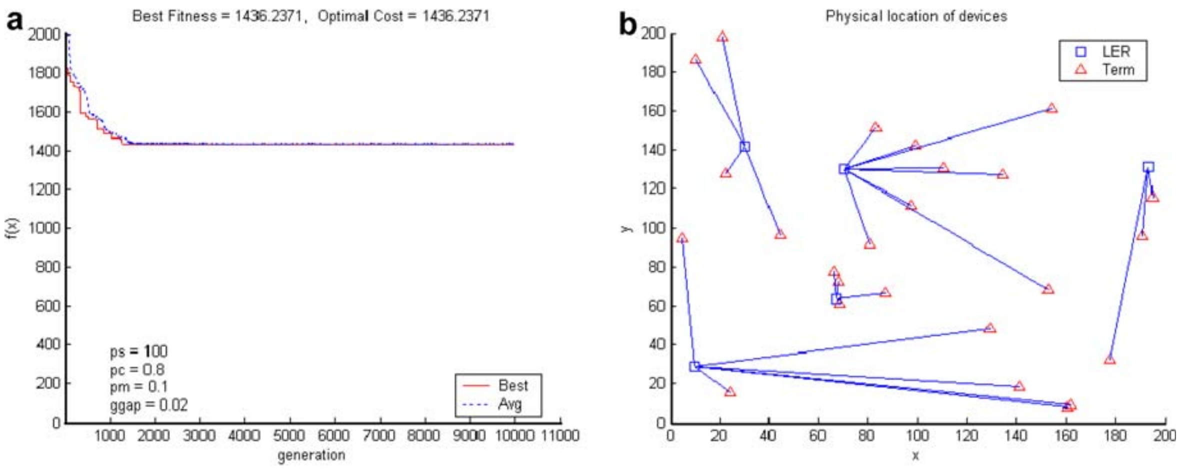


Fig. 4. Using GATA and discard infeasible solutions: (a) The changes of best and average fitness values vs. generations. (b) The best assignment found.

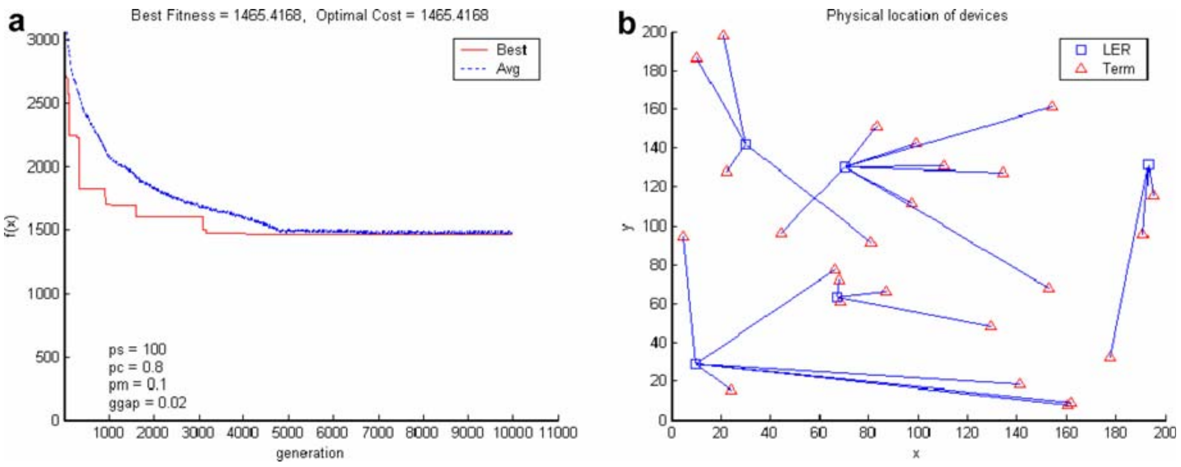


Fig. 5. Using GATA and a penalty function: (a) The changes of best and average fitness values vs. generations. (b) The best assignment found.

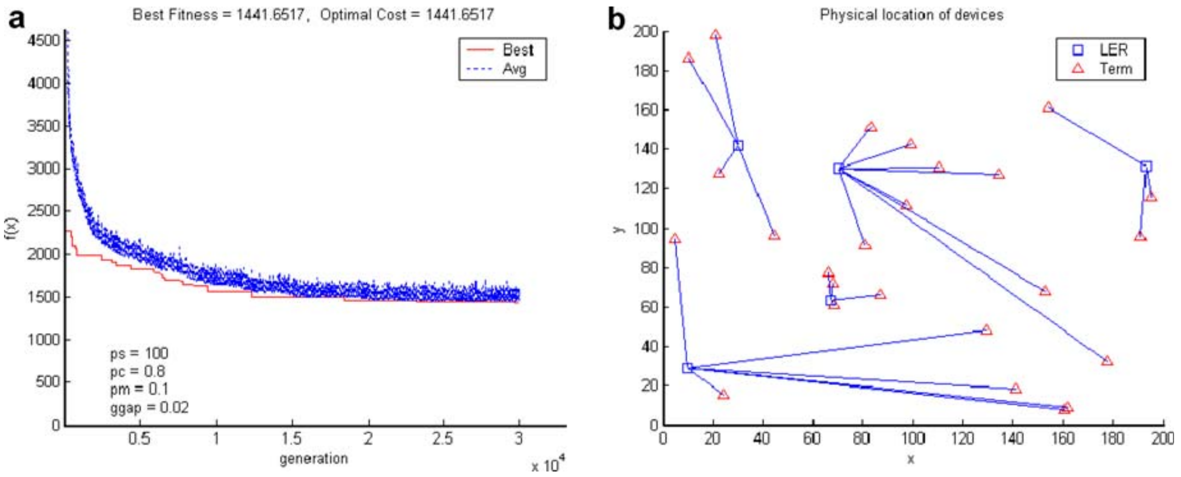


Fig. 6. Using GATA and another penalty function: (a) The changes of best and average fitness values vs. generations. (b) The best assignment found.

very close to optimal. Also a proper selection of the parameters can improve the results.

6.2. Access node locations

We carried out a number of experiments to test the performance of GANL by using a randomly generated network that has 20 possible access nodes and 100 terminals. The geographical locations of terminals and possible nodes are randomly generated on a 200×200 grid as shown in Fig. 7(a). The connection costs are computed as the Euclidian distance between the terminal and access nodes. The access node installation costs are generated randomly to be between 50 and 200. Using BB method, an optimal solution is found to be as shown in Fig. 7(b) with optimal cost of 4328.1345. Using GANL with $p_s = 500$, $p_c = 0.8$, $p_m = 0.02$ and $ggap = 0.1$, the best solution that satisfies the constraints and has minimal cost is shown in Fig. 8 with cost equal to 4504.045. We can clearly see that it is only 4% higher than the optimal cost.

6.3. Transit node locations and connectivity

To test our approach for solving TNLC, we considered a hypothetical backbone network with demand requirements as shown in Fig. 9. It is required to identify the location of transit nodes, link connectivity, link loads and the demand realization vector $\boldsymbol{x}$. This problem is formulated as a mixed-integer program (MIP) and solved using LINGO [25]. An optimal solution is found to be as illustrated in Fig. 10(a) with objective function value of 682. Then we carried out the solution using GATNLC with the following parameter settings: $p_s = 200$, $p_c = 0.8$, $p_m = 0.1$ and $ggap = 0.05$. In this experiment, only solutions that satisfy all inequality constraints and approximately the equality (demand) constraints are allowed to

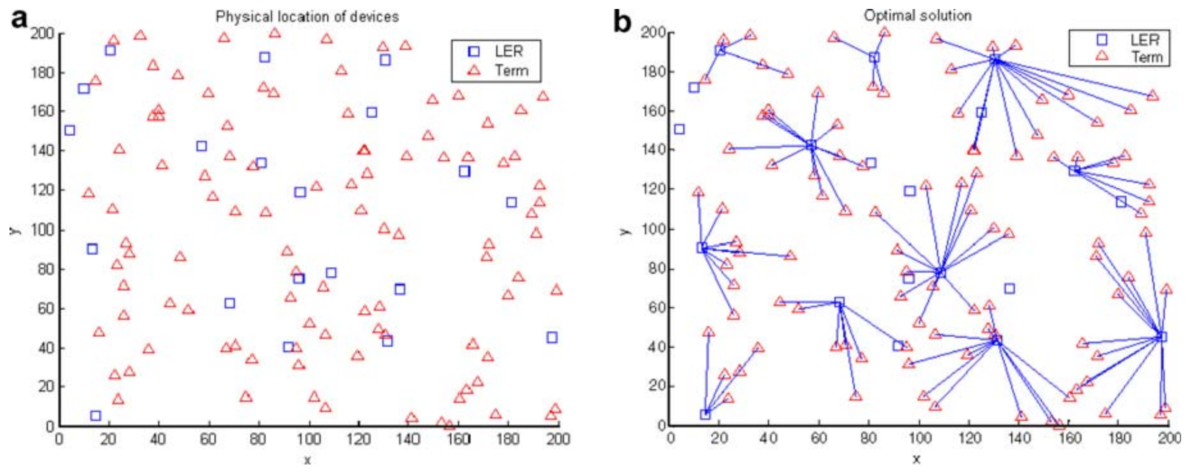


Fig. 7. ANL problem (a) Randomly generated terminal and possible node locations. (b) An optimal solution using BB method.

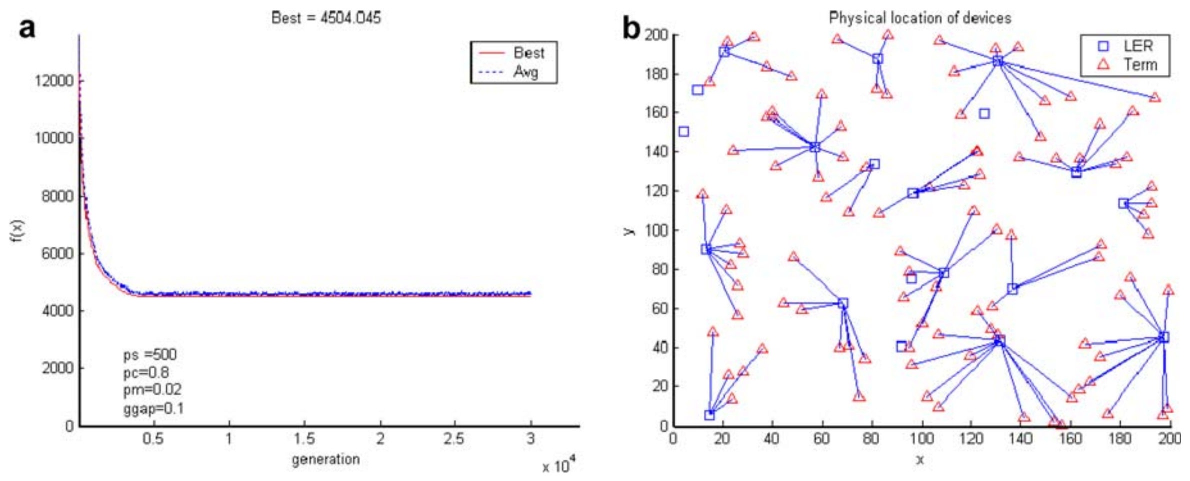


Fig. 8. Using GANL: (a) Changes of best and average fitness vs. generations. (b) The best solution found.

$N = 3,\ M = 4$
$\mathbf{h} = (h_1,\ h_2,\ h_3) = (10,\ 5,\ 8)$
$b_e = 15,\ e = 1,\ 2,\ \ldots,\ 11$
$k_v = 4,\ v = 1,2,\ \ldots,\ 4$
$f_e = 2,\ e = 1,\ 2,\ \ldots,\ 11$
$c_e = 10,\ e = 1,\ 2,\ \ldots,\ 11$
$\eta_v = 10,\ v = 1,2,\ \ldots,\ 4$
h_1: e_3-e_5-e_7, e_3- e_4- e_{10}, e_2- e_4- e_9, e_2- e_5- e_9- e_{11}
h_2: e_1- e_2, e_1-e_3- e_6
h_3: e_1- e_4- e_9, e_1- e_5- e_8

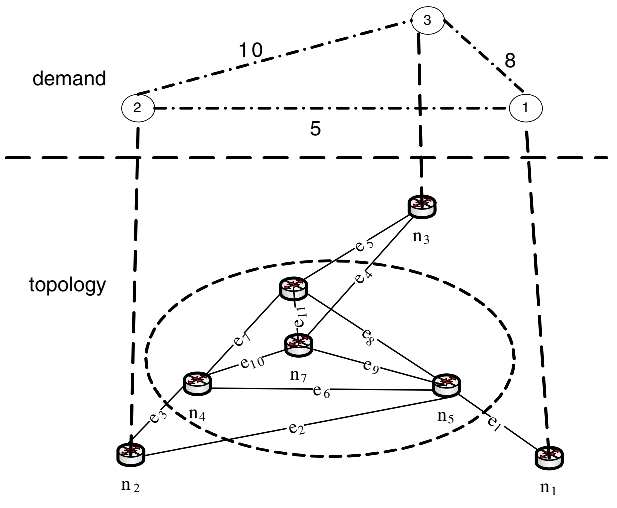
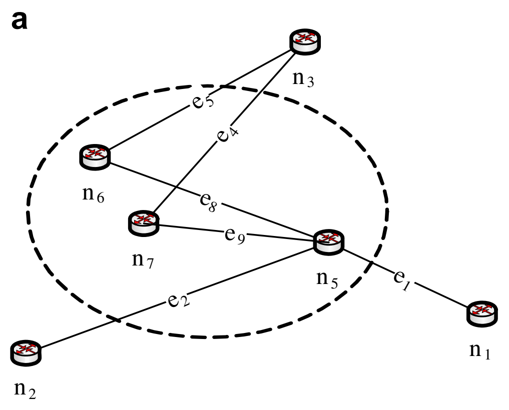


Fig. 9. A hypothetical backbone network.

be in the population. Using penalty functions as defined in Section 5.3 to penalize infeasible solutions, the variations of the best and average fitness values are shown in Fig. 10(b) and upon termination the best solution corresponds to an objective function value of 696 which is 2.1% higher than the optimal.



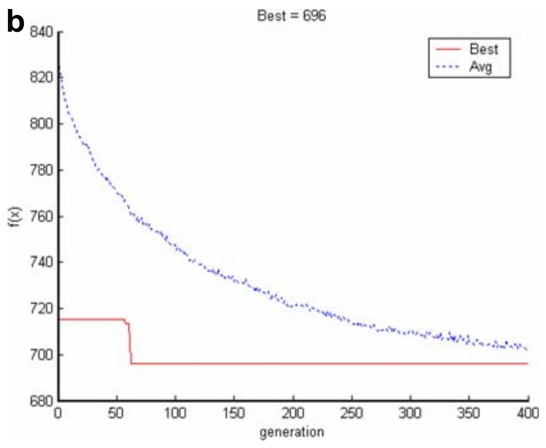


Fig. 10. (a) An optimal solution using LINGO. (b) Changes of best and average fitness vs. generations using GATNLC.

7. Conclusions and future work

In this study, the application of genetic algorithms to the topological design of MPLS-based networks is demonstrated. The proposed approach is tested using a number of hypothetical randomly generated networks. The simulation results show that the proposed method is effective and can produce optimal or close-to-optimal solutions. The accuracy of the solution obtained is affected by the GA parameter settings. Deciding on proper parameter settings is problem-dependent and is still open for research. Another important problem that needs to be investigated, especially when the entire hierarchical network is owned and operated by one institution, is the unified model. Under this model, the consideration of the dependence between different hierarchical levels during the design can lead to more cost-effective solutions. Finally, a detailed comparative study of various heuristic approaches for the design of hierarchical networks is worthwhile.

Acknowledgments

The author thank King Fahd University of Petroleum and Minerals (KFUPM) for their support during this work. Also special thanks due to the anonymous reviewers for their valuable comments that helped in enhancing the presentation of this manuscript.

References

[1] E. Rosen, A. Viswanathan, R. Callon, Multiprotocol label switching architecture, RFC 3031, January 2001.
[2] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, X. Xiao, Overview and principles of internet traffic engineering, RFC 3272, May 2002.
[3] R.R. Boorstyn, H. Frank, Large-scale network topological optimization, IEEE Transactions on Communications 25 (1) (1977) 29–47.
[4] T. Thomadsen, J. Clausen, Hierarchical network design using simulated annealing, Technical Report, Informatics and Mathematical Modelling, Technical University of Denmark, 2002.
[5] M. Schwartz, Computer Communication Network Design and Analysis, Prentice Hall, 1977.
[6] M. Pioro, D. Medhi, Routing, Flow, and Capacity Design in Communication and Computer Networks, Elsevier Inc., 2004.
[7] S. Srivatsa, P. Seshaiah, On the topological design of a computer network, Computer Networks and ISDN Systems 27 (4) (1995) 567–569.
[8] R. Dionne, M. Florian, Exact and approximate algorithms for optimal network design, Networks 9 (1979) 37–59.
[9] D. Whitley, Genetic Algorithms and Evolutionary Computing, Van Nostrands Scientific Encyclopedia, 2002.
[10] M. Srinivas, L.M. Patnaik, Genetic algorithms: a survey, IEEE Computer 27 (6) (1994) 17–27.
[11] M. Gen, R. Cheng, Genetic Algorithms and Engineering Design, Wiley, 2000.
[12] Z. Michalewicz, Genetic algorithms, numerical optimization, and constraints, in: Proc. 6th Int. Conf. Genetic Algorithms, Pittsburgh, July 15–19, 1995.
[13] S. Khuri, T. Chiu, Heuristic algorithms for the terminal assignment problem, in: Proc. 1997 ACM/SIGAPP Symp. Applied Computing, ACM Press, 1997.
[14] R. Elbaum, M. Sidi, Topological design of local-area networks using genetic algorithms, IEEE/ACM Transactions on Networking 4 (5) (1996) 766–778.
[15] H. Youssef, S. Sait, S. Khan, An evolutionary algorithm for network topology design, in: Proc. Int. Joint Conf. Neural Networks, vol. 1, 2001, pp. 744–749.
[16] J. Arabas, S. Kozdrowski, Applying an evolutionary algorithm to telecommunication network design, IEEE Transactions on Evolutionary Computation 5 (4) (2001) 309–322.
[17] H. Chou, G. Premkumar, C.-H. Chu, Genetic algorithms for communications network design – an empirical study of the factors that influence performance, IEEE Transactions on Evolutionary Computation 5 (3) (2001) 236–249.
[18] Z. Qin, F. Wu, N. Law, Designing B-ISDN network topologies using the genetic algorithm, in: Proc. 5th Int. Symp. Modeling, Analysis, and Simulation on Computer and Telecommunication Systems, 1997, pp. 140–145.
[19] H. Sayoud, K. Takahashi, B. Vaillant, Designing communication network topologies using steady-state genetic algorithms, IEEE Commununication Letters 5 (3) (2001) 113–115.
[20] M. Pioro, A. Myslek, A. Juttner, J. Harmatos, A. Szentesi, Topological design of MPLS networks, Globecom, San Antonio, 2001.
[21] A. Myslek, Greedy randomized adaptive search procedures (GRASP) for topological design of MPLS networks, in: Polish Teletraffic Symposium, Zakopane, 2001.
[22] E. El-Alfy, MPLS network topology design using genetic algorithms, in: Proc. 4th ACS/IEEE Int. Conf. Computer Systems and Applications, Dubai/Sharjah, UAE, March 2006.
[23] A. Chipperfield, P. Fleming, H. Pohlheim, C. Fonseca, Genetic Algorithms Toolbox for MATLAB, User's Guide.

[24] E. El-Alfy, Applications of genetic algorithms to MPLS-based network design, Technical Report, KFUPM-CCSE-2005-005/ICS, September 2005.
[25] LINGO User's Guide Manual, LINDO Systems Inc., 2004.

El-Sayed M. El-Alfy received his Ph.D. degree in Computer Engineering from Stevens Institute of Technology (SIT), USA in July 2001. He also holds M.S. in Computer Science from SIT (2001), as well as M.S. in Automatic Control: Intelligent Systems (1994) and B.S. (excellent with honor degree) in Electronics Engineering: Computer and Control Engineering (1991) from El-Mansoura University, Egypt. He is currently working as an Assistant Professor at the College of Computer Sciences and Engineering, King Fahd University of Petroleum and Minerals, Saudi Arabia since 2004. Before that he joined the Department of Computer and Control Engineering at Tanta University, Egypt as an Assistant Professor in 2002. He served as an IT Consultant at the Canadian Project, El-Mansoura, Egypt (2002-2004). Also, he was with Lucent Technologies, Inc. as a Member of Technical Staff (MTS) and a post-doctoral fellow at SIT (2001-2002), an Affiliated Instructor at SIT (1998-2001), a Graduate and Teaching Assistant at the Department of Computer and Control Engineering, Tanta University, Egypt (1992-1997), and a Research Assistant at the Department of Computer and Control Engineering, El-Mansoura University, Egypt (1991-1992). He is a member of IEEE, IEEE Communications Society CISTC, Arab Computer Society (ACS), and the Egyptian Body of Engineers. He is serving on the program committee of IEEE ICC 2007, IEEE CEC 2007, and ICIT 2007 conferences and as a reviewer for several conferences and some journals. His research interests include network design and performance optimization; security management; applications of computational intelligence in network-related problems and decision support systems.