26 NETWORK RELIABILITY OPTIMIZATION

Abdullah Konak[1] and Alice E. Smith[2]

[1]Information Sciences and Technology
Penn State Berks
Reading, PA 19610 USA
konak@psu.edu

[2]Industrial and Systems Engineering
Auburn University
Auburn University, AL 36849 USA
aesmith@eng.auburn.edu

Abstract: This chapter presents design of reliable networks. The exact calculation of any general network reliability measure is NP-hard. Therefore, network designers have been reluctant to use reliability as a design criterion. However, reliability is becoming an important concern to provide continuous service quality to network customers. The chapter discusses various network reliability measures and efficient techniques to evaluate them. Two genetic algorithms are presented to demonstrate how these techniques to estimate and compute network reliability can be incorporated within an optimization algorithm. Computational experiments show that the proposed approaches significantly reduce computational effort without compromising design quality.
Keywords: Network reliability, network resilience, network design, network survivability.

26.1 INTRODUCTION

While planning a telecommunication network, several competing interests such as cost, throughput, performance, connectivity requirements, and reliability must be considered. Among them, reliability has become an important concern in recent decades. Many new telecommunication technologies such as fiber-optic cables and high capacity switches have provided economical benefits by means of capacity concentration (Ball et al., 1995). As a result, telecommunication networks tend to be sparser com-

pared to networks based on traditional copper cables (Balakrishnan et al., 1998). A high capacity sparse network, however, is vulnerable to component (links and nodes) failures. Even a single component failure can significantly disturb the service quality of a network or leave many customers disconnected. Therefore, as the dependence on telecommunication networks increases and network topologies become sparser, network reliability becomes an important concern while designing new networks.

In the most general form, network reliability describes the ability of a network to continue network services in the case of component failures. This chapter focuses on designing reliable networks. The most challenging aspect of this problem is computing a reliability measure of a network. The exact calculation of most reliability measures is NP-hard (Ball, 1980). Therefore, an overwhelming body of research on network reliability has focused on developing efficient techniques to evaluate network reliability, including exact methods, theoretical bounds, and simulation. However, work on optimal reliable network design did not fully exploit or implement these efficient techniques in an optimization framework.

The chapter is organized as follows: Section 26.2 presents network reliability modeling and major reliability measures. Efficient evaluation of network reliability is very important for optimal network design. Therefore, the methods to evaluate network reliability measures are given in Section 26.3. Existing work on network reliability optimization is summarized in Section 26.4. Sections 26.5 presents a genetic algorithm (GA) to design reliable networks with an emphasis on demonstrating use of efficient reliability evaluation in a search algorithm. Section 26.6 presents a bi-objective GA to design resilient networks.

26.2 NETWORK RELIABILITY MODELING

Telecommunication networks consist of imperfect components. Both the links and the nodes of a network are subject to failure. Failure mechanisms of network components, especially those of links, have not been well defined in the literature (Ball et al., 1995). Nonetheless, failure rates can be derived from historical data. A telecommunication network with unreliable components is usually modeled as an undirected probabilistic network $G = (E,V)$ with node set V and arc set E such that each arc can be in either of two states: operative or failed, with associated probabilities p_{ij} and $1 - p_{ij}$, respectively. In this model, nodes represent telecommunication devices such as routers, switching stations, and computers, and arcs represent links connecting these devices. Although p_{ij} can be interpreted differently, it is usually defined as the probability that arc (i, j) is in the operative state at a random point in time. The common assumptions of this model are:

- Arc failures are independent;
- Nodes are perfectly reliable;
- No repair is allowed.

Hence, the probability of observing a particular state of the network is as follows:

$$Pr\{\mathbf{X}\} = \prod_{(i,j)\in E} [1 - p_{ij} + x_{(i,j)}(2p_{ij} - 1)] \tag{26.1}$$

where $x_{(i,j)} = 1$ if arc (i, j) is operative in state $\mathbf{X}$, $x_{(i,j)} = 0$ otherwise.

The assumptions of independent component failures and perfectly reliable nodes have been criticized as being impractical. Although in practice component failures are often observed together, the assumption of independent component failures is very important for computational tractability. Assuming perfectly reliable nodes is not very appropriate for telecommunication networks, either. However, a probabilistic network with unreliable nodes can be transformed to a probabilistic network with perfectly reliable nodes, or node failures can be incorporated into reliability calculations in some cases (Colbourn, 1987). Therefore, most network reliability analysis and optimization papers assume perfectly reliable nodes. In Section 26.6, we relax the assumption of perfectly reliable nodes and use a reliability measure in which node failures are considered.

26.3 NETWORK RELIABILITY MEASURES

Generally, three major types of network reliability measures are considered in the network reliability literature: connectivity, resilience, and performability measures. The majority of the research on network reliability focuses on connectivity measures since the primary function of a telecommunication network is to provide connectivity. With respect to connectivity, network reliability is expressed as the probability that a specified set of nodes (T) of the network are connected at a random point in time. This probability can be computed as follows:

$$R = \mathrm{E}[\Phi(\mathbf{X})] = \sum_{\mathbf{X} \in S} Pr\{\mathbf{X}\}\Phi(\mathbf{X}) \tag{26.2}$$

where S is the state space of all possible network states, and $\Phi(\mathbf{X})$ is a structure function defined as follows:

$$\Phi(\mathbf{X}) = \begin{cases} 1, & \text{if all nodes in } T \text{ are connected in state } \mathbf{X}; \\ 0, & \text{otherwise.} \end{cases} \tag{26.3}$$

The primary network reliability measures for undirected probabilistic networks are as follows

- *Two-terminal.* Probability that a selected node pair, a source node s, and a sink node t, are connected (i.e., $T = \{s,t\}$).
- *All-terminal.* Probability that every node can communicate with every other node in network G (i.e., $T = V$).
- *K-terminal.* Given a node set $K \subset V$, probability that every node in K can communicate with every other node in K (i.e., $T \subset V$).

A two-terminal measure is used when the communication between two specified nodes of a network is critical (e.g., two metropolitan areas, a file and a web server on different sites). The all-terminal measure is frequently used for the backbone level of packet switched networks since if a path fails in these networks, traffic can be rerouted around alternative paths as long as the network is globally connected. A K-terminal

measure is used when connectivity of a subset of network nodes is concern (e.g., virtual local area networks and distributed computing applications). The reliability measures defined above are strongly related with each other. A method that can be applied to one can also be applied to the others. Exact calculation of these reliability measures is NP-hard for general networks (Ball, 1980).

Network resilience measures, in fact, are special cases of connectivity measures. The connectivity measures given above assume that a network is not operational if the desired connectivity is lost. In practice, however, a network continues to serve the remaining connected components even though one or more nodes have become disconnected. To evaluate the ability of a network coping with catastrophic failures and recovering network services in disconnected states, several network resilience measures have been proposed such as the probability that all operative node pairs can communicate and the expected fraction of node pairs communicating (Ball, 1979). Unlike the binary structure function in the connectivity based reliability measures, the structure function is multi-model in resilience measures, which usually makes their calculation harder than reliability measures.

Performance metrics such as response time and throughput are commonly used in design and analysis of telecommunication networks. Network performability measures are concerned with the performance of a network in various network states. Given a performance metric of a network (Ω), performability measures are usually expressed in three cases:

- $Pr\{\Omega \leq \alpha\}$;
- $E[\Omega]$;
- $E[\Omega \mid$ at most k components fail$]$.

The last group of the performability measures is primarily used to reduce the computational complexity by using a k value of usually one or two. The justification for this assumption is that network components are usually so reliable that it is adequate in practice to consider states with no and single failure only since these states cover most of the state-space probability (Sanso et al., 1992).

26.4 EVALUATION OF RELIABILITY

26.4.1 Exact Methods

Methods for exact reliability calculation are two-fold: cut/path set enumeration methods and state enumeration methods (Ball et al., 1995). Cut/path set methods require enumerating all cut/path sets of a network. For example, given all path sets of a network, $P_1, \ldots, P_h$, let E_i be the event that all arcs in P_i are operational, then network reliability is calculated as

$$R(G) = Pr\{E_1 \cup E_2 \cup \ldots \cup E_h\} \tag{26.4}$$

Unfortunately, equation (26.4) cannot be calculated easily since the E_i's are not mutually exclusive events and the number of path sets of a network is exponential

in the number of arcs. In fact, a naive implementation of equation (26.4) leads to a doubly exponential time algorithm (Ball et al., 1995). For example, the backtracking algorithm (Ball and Slyke, 1977), which is based on generating cut sets, could be used in Deeter and Smith (1998) to design networks with only 5 nodes. The domination theorem developed by Satyanarayana and Prabhakar (1978) provides substantial improvement in the computation of equation (26.4). The most efficient algorithms based on cut/path sets are the algorithms proposed by Ball and Nemhauser (1979) and Provan and Ball (1984), which can compute reliability in polynomial time in the number of minimal path and cut sets.

The most basic state based method is complete state enumeration, requiring the generation of all 2^m states of a network with m arcs. Applying reliability preserving network reductions and network decomposition techniques can significantly reduce the computational effort in state space enumeration. In network reductions, a network G is reduced to a network G' with fewer nodes and/or arcs such that

$$R(G) = \lambda R(G') \tag{26.5}$$

where λ is a reliability-preserving multiplicative constant depending on reductions. In network decomposition, a network is partitioned into two or more subnetworks; then, the reliabilities of these subnetworks are calculated and used to obtain the reliability of the original network. This technique is successfully used for two-terminal reliability in directed networks (deMercado et al., 1976; Singh and Ghosh, 1994; Hagstrom, 1984). A powerful network decomposition technique is to pivot the reliability expression on the state of an individual arc, which is also known as factoring (Colbourn, 1987; Page and Perry, 1991; Resende, 1986; 1988; Satyanarayana and Chang, 1983). Conditioning on the state of arc (i, j), the reliability expression for $R(G)$ is given by

$$R(G) = R(G \cdot (i,j))p_{ij} + R(G - (i,j))(1 - p_{ij}) \tag{26.6}$$

where $G \cdot (i, j)$ is a network obtained from G by merging nodes i and j into a new node and connecting each arc incident to them to this new node, and $G - (i, j)$ is a network obtained by deleting arc (i, j) from G. If this decomposition is repeatedly applied with proper arc selection for pivoting and network reductions, significant improvements can be achieved in reliability computation since only relevant states of the network are considered (Satyanarayana and Chang, 1983). In reality, any technique to compute reliability requires exponential time in the worst case. For undirected networks, factoring with network reductions provides the best possible time (see (Ball et al., 1995) for a comprehensive discussion). In Section 26.5, we used a factoring algorithm to calculate the all-terminal measure.

26.4.2 Bounds

Because of the intractability of exact reliability calculation, theoretical bounds on reliability were used as a substitute for actual reliability in several network reliability papers (Jan, 1993; AboElFotoh and Al-Sumait, 2001; Dengiz et al., 1997a;b). Bounds for approximating network reliability can be considered in three groups:

- Bounds based on the reliability polynomial (Slyke and Frank, 1972; Ball and Provan, 1983);

- Bounds based on arc packing by cut or path sets (Lomonosov and Polesskii, 1972; Aboelfotoh and Colbourn, 1989; Brecht and Colbourn, 1988);
- Bounds based on the most probable states (Li and Silvester, 1984; Lam and Li, 1986; Yang and Kubat, 1989).

The first group of bounds depends on counting a small fraction of operational network states to approximate reliability. An important shortcoming of this approach is that it is applicable only to networks with identical arc reliabilities. The second group of bounds considers only a fraction of all possible cut/path sets of a network to obtain a bound. These bounds can be used for networks with arbitrary arc reliabilities. However, they are computationally demanding since they require generating cut/path sets or an effective arc packing of a network. In addition, these bounds can be used only for connectivity based reliability measures with a binary structure function.

The most probable state bounds are based on the observation that the reliabilities of individual components are usually so high that only a small fraction of all possible states will cover the majority of the probability. Although the number of the possible network states is enormous, most of them have extremely low probabilities of occurrence; hence, they can be ignored. The most probable states method requires enumerating k most probable states of a network, $\mathbf{X}^1$, $\mathbf{X}^2$,...,$\mathbf{X}^k$, such that $Pr\{\mathbf{X}^1\} \geqslant Pr\{\mathbf{X}^2\} \geqslant, \ldots, \geqslant Pr\{\mathbf{X}^k\}$ and $Pr\{\mathbf{X}^k\} \geqslant Pr\{\mathbf{X}^i\}$ for all remaining states i of the network. Upper and lower bounds on reliability based on the k most probable states are given as follows:

$$R_U(G) = \sum_{i=1}^{k} \Phi(\mathbf{X}^i) Pr\{\mathbf{X}^i\} + (1 - \sum_{i=1}^{k} Pr\{\mathbf{X}^i\})$$

$$R_L(G) = \sum_{i=1}^{k} \Phi(\mathbf{X}^i) Pr\{\mathbf{X}^i\}$$

The tightness of these bounds depends on the number of the states considered. Several methods have been proposed to efficiently enumerate the k most probable states of a network (Li and Silvester, 1984; Lam and Li, 1986; Yang and Kubat, 1989).

26.4.3 Simulation and other Estimation Techniques

Simulation has been a major alternative to estimate new reliability, especially for large networks, due to the intractability of the exact calculation of network reliability and the absence of tight bounds. The application of simulation to estimate new reliability and performability measures is outwardly straightforward. For example, the procedure for the Crude Monte Carlo (CMC) method, which is based on sampling of network states, is given as follows:

Step 1. Set $R = 0$
Step 2. For $k = 1 \ldots K$ do following steps

a. For each $(i, j) \in E$ generate a random number U, then if $p_{ij} \leq U$ then $x_{(i,j)} = 1$ else $x_{(i,j)} = 0$

b. $R = R + \Phi(\mathbf{X})$

Step 3. $\widehat{R}(G) = R/K$

Although the CMC simulation is easy to implement, reliability estimation using simulation is computationally very expensive. A major concern is that networks and components are usually so reliable that large numbers of network states are required to be sampled in order to observe a few network failures and obtain an accurate estimate of reliability. This significantly increases the cost of accurate estimations especially for highly reliable networks. An accurate estimate is particularly important if simulation is to be used within an optimization algorithm to compare alternative designs. Several alternative approaches such as dagger sampling (Kumamoto et al., 1980), stratified sampling (Slyke and Frank, 1972), the Markov model (Mazumdar et al., 1999), the sequential construction/destruction methods (Easton and Wong, 1980; Fishman, 1986a), importance sampling using bounds (Fishman, 1986b), graph evolution method (Elperin et al., 1991) and sampling based on failures sets (Kumamoto et al., 1977) have been proposed in the literature to improve efficiency and effectiveness of simulation in estimating network reliability.

As a new approach to estimate network reliability, Srivaree-ratana et al. (2002) used artificial neural networks to estimate all-terminal and two-terminal reliability.

26.5 RELIABLE NETWORK DESIGN PROBLEM

26.5.1 Problem Formulation

In the most general form, the optimization problem is to find a network topology maximizing the reliability for a given cost constraint or minimizing the design cost for a given reliability constraint. We refer to these two problems as P1 and P2, respectively. As a part of the overall network design problem, however, P2 is a more common problem. In addition to the cost or reliability constraint, there might be some other constraints restricting topologies such as survivability constraints (e.g., node degrees, node or link connectivity requirement) or performance constraints (e.g., network diameter).

Decision variable z_{ij} represents the type of the arc installed between nodes i and j from a discrete set of arc types $\{0,1,...,L\}$ where L is the most reliable arc type and $z_{ij} = 0$ means that no arc is installed on (i, j). Then, the design cost is:

$$C(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{ij}(z_{ij}) \tag{26.7}$$

where $c_{ij}(l)$ is the cost of installing arc type l between nodes i and j. This cost usually depends on the distance between node pairs and may include fixed and variable costs such as cabling, installation costs, and right of way costs. In many formulations of the problem, arc types are ignored and only binary decision variables z_{ij} are used.

The problem formulated in this chapter is to minimize the design cost subject to a given minimum all-terminal reliability requirement R_{min} and a 2-node connectivity constraint. A network is 2-node connected if at least 2 nodes must be removed in order to disconnect the remaining nodes. For fiber-optic networks, 2-node/link connectivity

is frequently used to ensure a minimum level of survivability (Monma and Shallcross, 1989).

26.5.2 Solution Approaches

Alternative methods have been proposed in the literature to solve the network design problem considering reliability. Due to the difficultly of the problem and calculating reliability measures, however, few exact methods were developed. Aggarwal et al. (1982a;b) proposed an approach based on enumeration of the spanning trees of the possible network topology. However, this approach is only applicable to very small networks. Jan et al. (1993) developed a branch-and-bound algorithm to minimize the design cost subject to a minimum all-terminal reliability constraint. In this approach, the problem was sequentially solved by considering solutions with only $n-1$, n, $n+1$,...,$n(n-1)/2$ arcs. An upper bound on reliability was used to efficiently eliminate infeasible branches, and a lower bound was used to fathom unpromising subproblems. The approach was tested for networks up to 12 nodes. However, this approach is only applicable to networks with identical arc failure probabilities. In another paper, Jan (1993) investigated the topological features of networks that may lead to high reliability.

Belovich (1995) developed a construction heuristic to enhance the reliability of an existing network by adding the most promising arcs to the network. The reliability metric used in this study was the probability that all packets arrive at their destination, which is estimated by a linear-time upper bound.

Kumar et al. (1995a;b) developed a GA to design and extend existing networks considering various objectives such as reliability, delay, and average nodal distance. Genetic Algorithm

Dengiz et al. (1997a;b) presented a GA approach for the network design problem to maximize all-terminal reliability under a cost constraint. An upper bound on reliability was used to evaluate the fitness of candidate solutions, and simulation was used to estimate the reliability of the best solution. In their GA, the crossover operator was supported with a network repair algorithm to make sure that offspring had a minimum node degree of two. Due to the limitation of the upper bound used in their GA, only arcs with identical reliability were assumed. Deeter and Smith (1998) also proposed a GA to solve both P1 and P2 with different arc types and objective functions (two-terminal and all-terminal reliability). In their method, network reliability was exactly calculated for small size networks and CMC simulation was used for larger networks.

AboElFotoh and Al-Sumait (2001) developed a neural network (NN) approach to solve the problem. In this approach, each arc was represented by a neuron. If an arc was selected in a solution, the corresponding neuron's output became one (i.e., the neuron fired), otherwise zero. The NN aimed to minimize an energy function composed of three terms: the cost of a solution, reliability, and a penalty term for higher reliability than required. To evaluate the reliability term in the energy function, an upper bound was used when arc reliabilities were high or a lower bound was used when arc reliabilities were low. Srivaree-ratana et al. (2002) used a NN approach to estimate all-terminal reliability within a simulated annealing algorithm to evaluate candidate solutions.

26.5.3 A Genetic Algorithm to Design Reliable Networks

In this section, a GA is introduced to solve the formulated problem. The main difference between the GA herein and previous approaches is that the GA uses a combination of techniques for efficient evaluation of all-terminal reliability so that large problems with practical importance can be solved. Unlike some of the previous approaches, the reliability evaluation procedure of the GA is efficient for general types of networks. In addition, the GA employs advanced operators to deal with the 2-node connectivity survivability constraint.

26.5.3.1 Problem Encoding. A node-to-node adjacency matrix representation with arc types is used to represent solutions. In this representation, a network is stored in an $n \times n$ matrix, $Z = \{z_{ij}\}$, such that $z_{ij} = l$ if arc type l is used between nodes i and j, $l = 0,...,L$ where type 1 denotes the least reliable, type L denotes the most reliable arc type, and type 0 means that no arc exists between nodes i and j. If arc types are not considered , then $L = 1$ with $z_{ij} = 1$ denoting that arc (i, j) is selected in the solution, otherwise $z_{ij} = 0$.

26.5.3.2 Crossover Operator. The GA's crossover operator is basic uniform crossover with an efficient repair algorithm for 2-node-connectivity as follows:

Step 1. Randomly select two parents $X = \{x_{ij}\}$ and $Y = \{y_{ij}\}$
Step 2. Set $z_{ij} = \lfloor 1 - U \rfloor x_{ij} + \lceil U \rceil y_{ij}$ for $i = 1,...,n$, $j = i+1,...,n$
Step 3. **If** $Z \subseteq X$ or $Z \subseteq Y$ **Then** return Z and stop.
Step 4. For each node $v = 1,...,n$, perform the following steps.

- Step 4.1 Delete node v from offspring Z, and let s be the node with smallest index in $V \setminus v$.
- Step 4.2 $S(v) = \{s\}$, LIST=$\{s\}$, $\overline{S}(v) = V \setminus v$
- Step 4.3 Select node i from the end of LIST.
- Step 4.4 **If** there exists an arc (i, j) such that $j \in \overline{S}(v)$ **Then** $\overline{S}(v) = \overline{S}(v) \setminus j$, $S(v) = S(v) \cup j$, and LIST=LIST $\cup j$ **Else** LIST=LIST $\setminus i$
- Step 4.5 **If** LIST$\neq \emptyset$ **Then** go to Step 4.3
- Step 4.6 **If** $\overline{S}(v) = \emptyset$ **Then** stop since Z is 2-node connected with respect to v.
- Step 4.7 Find the minimum distance arc (i, j) such that $i \in S(v)$, $j \in \overline{S}(v)$, and $x_{ij} + y_{ij} \geq 1$, and set $z_{ij} = INT(1, L)$.
- Step 4.8 $\overline{S}(v) = \overline{S}(v) \setminus j$, $S(v) = S(v) \cup j$, and LIST=LIST $\cup j$
- Step 4.9 Go to Step 4.3

Step 5. Return Z

26.5.3.3 Mutation Operators. The GA has several mutation operators perturbing network topologies without disturbing 2-node connectivity. To achieve this, the mutation operators create a solution from an existing solution by changing the cycles of the existing solution. This approach was used by Monma and Shallcross (1989) to find minimum cost 2-node and 2-arc connected network topologies. The mutation operators are given below.

Procedure: One-Link-Exchange

Step 1. Find a random cycle C of at least four nodes.
Step 2. Randomly choose four nodes a, b, c, and d of C such that $z_{ab} \geq 1$, $(a,b) \notin C$, and $z_{cd} = 0$.
Step 3. Set $z_{cd} = z_{ab}$ and $z_{ab} = 0$.

Procedure: Two-Link-Exchange

Step 1. Find a random cycle C of at least four nodes.
Step 2. On cycle C, randomly determine two arcs (a,b) and (c,d) such that $z_{ab} \geq 1$, $z_{cd} \geq 1$, $z_{ac} = 0$, and $z_{bd} = 0$.
Step 3. Set $z_{ac} = z_{ab}$, $z_{bd} = z_{cd}$, $z_{ab} = 0$, and $z_{cd} = 0$.

Procedure: Three-Link-Exchange

Step 1. Find a random cycle C of at least four nodes.
Step 2. On cycle C, randomly choose three arcs (a,b), (c,d) , and (e,f) such that $z_{ad} = 0$, $z_{be} = 0$, and $z_{cf} = 0$.
Step 3. Set $z_{ad} = z_{ab}$, $z_{be} = z_{ef}$, $z_{cf} = z_{cd}$, $z_{ab} = 0$, $z_{cd} = 0$, and $z_{ef} = 0$.

Procedure: Add-Two-and-Remove-One-Arcs

Step 1. Find a random cycle C of at least four nodes.
Step 2. Randomly choose an arc (a,b) on cycle C such that $z_{ac} = 0$ and $z_{bd} = 0$.
Step 3. Set $z_{ac} = \min(1, z_{ab} - 1)$, $z_{bc} = \min(1, z_{ab} - 1)$, and $z_{ab} = 0$.

Procedure: Add-One-and-Remove-Two-Arcs

Step 1. Find two random adjacent cycles C' and C'' with a common arc (a,b).
Step 2. Remove arc $(a,c) \in C'$ and $(b,d) \in C''$, i.e. $z_{ac} = 0$ and $z_{bd} = 0$.
Step 3. Set $z_{cd} = \max(L, z_{ac} + 1, z_{bd} + 1)$.

Procedure: Delete-an-Arc

Step 1. Find a random cycle C of at least four nodes.
Step 2. Randomly choose two nodes a and b such that $z_{ab} \geq 1$ and $(a,b) \notin C$.
Step 3. Set $z_{ab} = 0$

Procedure: Add-an-Arc

Step 1. Randomly choose two nodes a and b such that $z_{ab} = 0$
Step 2. Set $z_{ab} = INT(1, L)$

Procedure: Change-Arc-Type

Step 1. Randomly choose an arc (a,b) such that $z_{ab} \geq 1$
Step 2. Change type of arc (a,b) to another type.

26.5.3.4 Fitness Function. The mutation and crossover operators of the GA always produce feasible network topologies with respect to the connectivity requirement. However, a candidate solution may violate the reliability constraint. Infeasible solutions with respect to reliability are penalized using a penalty function as follows

$$f(Z,t) = C(Z) + C_{\min}(t)\theta(t)\left(\frac{\max(0, R_{\min} - R(Z))}{R_{\min}}\right) \tag{26.8}$$

where $C_{min}(t)$ is the cost of the cheapest solution in the population, and $\theta(t)$ is an adaptive penalty factor, which is updated at each generation t as follows

$$\theta(t) = \theta(t-1)(0.5 + \text{fraction of infeasible solutions }) \tag{26.9}$$

Without requiring any parameter setting, this adaptive penalty function is a simplified version of the adaptive penalty function given by Coit and Smith (1996) and Coit et al. (1996). For the first generation, $\theta(1) = 1$, and then in the following generations it is increased if the population includes more infeasible solutions than feasible ones, or it is decreased if the otherwise is true.

26.5.3.5 Calculation of All-terminal Reliability. The reliability evaluation subroutines of the GA include an upper bound for quick assessment of all-terminal reliability, simulation (the Sequential Construction Method (Easton and Wong, 1980)), and an exact method based on the factoring procedure given by Page and Perry (1991) with minor modifications. The overall objective in the reliability evaluation is to minimize rigorous analysis of reliability without losing accuracy. The details of the reliability evaluation is given below.

When a new solution Z is produced, first an upper bound ($R_U(Z)$) on the reliability is calculated as follows:

$$R_U(Z) = 1 - \sum_{i=1}^{n}\left(\left(\prod_{k=1}^{n}(1-p_{ik})\right)\prod_{j=1}^{i-1}\left(1 - \frac{\prod_{k=1}^{n}(1-p_{jk})}{1-p_{ij}}\right)\right) \tag{26.10}$$

where $p_{ij} = 0$ if $z_{ij} = 0$. This bound can be used with arbitrary arc reliabilities, and in addition, it is computationally very efficient since only cut sets separating individual nodes, which can be identified easily, are considered. If $R_U(Z) < R_{min}$, solution Z is infeasible; therefore, $R(Z)$ is not evaluated further by simulation or factoring, and $R(Z) = R_U(Z)$ is used in the fitness calculation.

If the infeasibility of solution Z is not determined by the upper bound, then simulation or factoring is used to evaluate its reliability. However, if solution Z is not promising, meaning that it has a higher cost than the cost of the *Best Feasible Solution* found so far in the search, a rigorous analysis of reliability is not required. Therefore, the reliability of a non-promising solution is estimated by simulation using a very low number of replications. For a promising solution, factoring or simulation with a high number of replications is used depending on the size and density of the solution. For problems with 10 or less nodes, factoring is used regardless of the density of solutions. For problems with larger than 10 nodes, if $m \leq 1.5n$, factoring is used; otherwise, simulation is used to estimate the reliability.

When simulation is used, the estimated reliability is a random variable. Therefore, to ensure the feasibility of a promising solution Z with $100(1-\alpha)$ percent confidence, the reliability constraint is modified as follows

$$\widehat{R}(G) - z_\alpha \sigma_{\widehat{R}(Z)} \geq R_{min} \tag{26.11}$$

where $\widehat{R}(G)$ is the estimated all-terminal reliability, and $\sigma_{\widehat{R}(Z)}$ is the standard deviation of the estimation.

26.5.3.6 Overall Algorithm. The GA has a dynamic population size (μ), which is randomly and uniformly selected between μ_{min} and μ_{max} in each generation. In a generation, each solution participates in crossover exactly two times with randomly selected solutions. The mutation rate (MR) determines the probability of solutions being mutated in each generation. In mutation, one of the mutation operators is selected randomly and uniformly. Iterations continue until no new solution improving the Best Feasible Solution is found in the last gn_{max} solutions evaluated or a maximum number of g_{max} new solutions are evaluated. The details of the GA's overall procedure is given as follows:

Step 1. $t = 1, \theta(t) = 1\ \mu(t) = INT(\mu_{min}, \mu_{max})$
Step 2. Generate $\mu(t)$ initial solutions.
Step 3. **Crossover:** Do the following steps for $i = 1, ..., \mu(t) - 1$:

- Step 3.1 Crossover the i^{th} and $(i+1)^{th}$ solutions.
- Step 3.2 Evaluate and add offspring to the end of the population. Update the Best Feasible Solution if necessary.

Step 4. **Mutation:** Do the following steps for $i = 1, ..., \mu(t)$:

- Step 4.1 Generate a random number U. If $U \leq MR$ then randomly and uniformly select a mutation operator, and mutate solution the i^{th} solution.
- Step 4.2 Evaluate the mutated solution, update the Best Feasible Solution if necessary, and replace the original solution with the mutated one.

Step 5. Stop if $t \geq g_{max}$ or the Best Feasible Solution has not been updated in last gn_{max} solutions evaluated.
Step 6. Update $\theta(t)$, and calculate the fitness of the population.
Step 7. Sort the population in the descending order of the fitness.
Step 8. $t = t + 1$ and $\mu(t) = \min(\mu(t-1), INT(\mu_{min}, \mu_{max}))$
Step 9. Shuffle the first $\mu(t)$ solutions of the population, and delete the rest while making sure that the Best Feasible Solution stays in the population. Go to Step 3.

26.5.3.7 Computational Experiments. To test the performance of the GA, several problems from the literature are studied. The first set of problems includes 8, 9, 10, 15, 20, and 25-node test problems taken from Dengiz et al. (1997a). These problems have identical arc reliabilities (i.e., no arc choice is available); therefore, the decision variables are binary variables indicating whether to include an arc in a solution or not. In Dengiz et al. (1997a), a GA is used to solve the problems,and a branch-and-bound algorithm is also implemented to find optimal solutions for small problems. The second set of problems is taken from Deeter and Smith (1998), and includes a 10-node problem with three arc reliability choices of .70, .80, and .90 and a 19-node problem with three arc reliability choices of .96, .975, and .99.

For each problem instance, the GA was run 10 times using a different random number seed in each run. The computational results given in Tables 26.1 and 26.3

Table 26.1 Computational results for the problems with identical arc reliability.

					Breakdown of the Computational Effort in Percents			
n	p	R_{min}	Solutions Search	CPU per Solution	Factoring	Simulation $K = 200$	Simulation $K = 20,000$	Upper Bound
8	0.90	0.90	11059.0	0.003	4.67	95.33	0.00	0.00
8	0.90	0.95	11229.2	0.003	14.97	75.76	0.00	9.26
8	0.90	0.99	11440.9	0.003	0.12	30.05	0.00	69.83
8	0.95	0.90	10649.7	0.004	0.09	99.90	0.00	0.00
8	0.95	0.95	10526.0	0.004	0.22	99.78	0.00	0.00
8	0.95	0.99	10705.6	0.004	0.25	53.88	0.00	45.86
9	0.90	0.90	11729.9	0.004	4.15	95.84	0.00	0.00
9	0.90	0.95	11705.3	0.003	7.92	64.21	0.00	27.87
9	0.90	0.99	13936.3	0.002	0.19	32.22	0.00	67.58
9	0.95	0.90	12926.0	0.003	0.12	99.88	0.00	0.00
9	0.95	0.95	11438.4	0.004	0.25	99.74	0.00	0.00
9	0.95	0.99	13057.7	0.003	0.81	49.76	0.00	49.42
10	0.90	0.90	11078.1	0.003	1.93	98.06	0.00	0.00
10	0.90	0.95	12549.4	0.003	8.78	57.12	0.00	34.10
10	0.90	0.99	14008.4	0.003	0.20	32.08	0.00	67.72
10	0.95	0.90	11710.4	0.004	0.17	99.83	0.00	0.00
10	0.95	0.95	11564.5	0.004	0.24	99.76	0.00	0.00
10	0.95	0.99	13517.0	0.003	2.86	47.29	0.00	49.85
15	0.90	0.90	22951.1	0.004	36.48	56.64	0.03	6.84
15	0.90	0.95	21954.7	0.004	15.13	43.16	0.18	41.52
15	0.90	0.99	21604.2	0.005	0.00	41.45	0.96	57.60
15	0.95	0.90	16217.2	0.004	0.77	99.20	0.02	0.00
15	0.95	0.95	15167.7	0.004	9.10	90.87	0.03	0.00
15	0.95	0.99	19444.3	0.004	0.90	39.16	0.13	59.79
20	0.90	0.90	31640.3	0.004	32.21	50.12	0.04	17.62
20	0.90	0.95	26509.1	0.008	1.49	43.30	1.31	53.89
20	0.90	0.99	33876.6	0.011	0.00	42.20	2.37	55.43
20	0.95	0.90	22477.0	0.005	6.37	93.61	0.02	0.00
20	0.95	0.95	24829.2	0.005	19.14	80.84	0.02	0.00
20	0.95	0.99	26770.1	0.005	1.30	41.76	0.33	56.60
25	0.90	0.90	42792.1	0.012	31.08	44.59	0.14	24.19
25	0.90	0.95	42863.8	0.036	3.72	43.66	4.15	48.47
25	0.90	0.99	43971.7	0.001	0.00	37.85	1.30	60.85
25	0.95	0.90	31845.4	0.006	7.61	92.36	0.02	0.00
25	0.95	0.95	32155.1	0.006	24.84	75.11	0.03	0.01
25	0.95	0.99	37142.4	0.019	2.38	43.58	1.77	52.27

represent the averaged values over ten runs. The parameters of the GA used in all runs are as follows: $\mu_{min} = 25$, $\mu_{max} = 50$, $MR = 0.1$, $gn_{max} = 10,000$, and $g_{max} = 100,000$. In simulation, $K = 20,000$ is used for promising solutions and $K = 200$ for non-promising solutions.

Table 26.1 lists the computational effort to solve the problems with identical arc reliabilities. In the table, the total number of solutions evaluated is broken down into groups with respect to the reliability evaluation method used. For example, for the 8-node problem with $p = 0.90$ and $R_{min} = .99$, a total of 11,440.9 solutions were evaluated on the average over 10 runs, and factoring was used for .13 percent, simulation with 200 replications for 30.05 percent, and only the upper bound for 69.83 percent to evaluate the solutions. For most cases, factoring or simulation with $K = 20,000$ was used to evaluate a very small fraction of solutions searched. When $R_{min} = .99$, the upper bound was especially effective in identifying infeasible solutions, reducing the need for factoring or simulation. However, the upper bound did not provide useful information when R_{min} and p were low. The highest percent of rigorous evaluation occurred when $R_{min} = .90$. For these cases, however, the CPU time per solution did not increase significantly or even decreased in some cases. The main reason for this is that $R_{min} = .90$ can be achieved with sparse networks with a few arcs, and the reliability of these networks can be computed by factoring using only several pivots and network reductions. Although more solutions were exactly evaluated for these cases, each evaluation took significantly less time, in turn, improving overall CPU time per solution.

Table 26.2 summarizes the results for the problems with identical arc reliabilities. In this table, the cost and reliability of the Best Feasible Solution found, the average best cost over 10 runs, the difference between the cost of the best and the worst solutions found in 10 runs, and the p-value for Best Feasible Solution are given. Here, p-value is the probability that the reliability of the Best Feasible Solution is higher than R_{min}. A p-value of one indicates that all-terminal reliability was exactly computed. In most cases, the GA found the previously reported optimal solutions or improved upon the previous best results. As seen in the table, the GA was very robust over random number seeds. In many cases, the same solution was found in 10 runs or the best and worst solutions were very close. For most cases, the reliability of the Best Feasible Solution was exactly calculated by factoring.

Table 26.3 lists the computational results for the problems with arc choices. Compared with the computational results of Set I, a higher percent of solutions were rigorously evaluated. However, allowing multiple arc choices did not complicate the reliability calculation as reasonable CPU times were observed. Similarly, for high R_{min} values, the upper bound alone was quite useful in identifying infeasible solutions. As seen in Table 26.4, the GA improved upon previous results.

26.6 RESILIENT NETWORK DESIGN PROBLEM

26.6.1 Problem Definition

In Section 26.3, network resilience measures were briefly introduced. In this section, a network resilience measure, called traffic efficiency (T), is used to design resilient networks. The traffic efficiency of a network G is given as

$$T(G) = \frac{1}{\gamma} \sum_{\mathbf{X} \in S} \left(\sum_{i=1}^{n} \sum_{j=i+1}^{n} \tau_{ij}(\mathbf{X}) t_{ij} \right) P\{\mathbf{X}\} \tag{26.12}$$

Table 26.2 Results for the problems with identical arc reliabilities.

			Best	Results of the Genetic Algorithm				
			Solution	Best	Mean		Best	
n	p	R_{min}	Reported	Cost	Cost	Range	Reliability	p-Value
8	0.90	0.90	203*	208.00	208.00	0	0.931722	1.0
8	0.90	0.95	247*	247.00	247.00	0	0.961377	1.0
8	0.90	0.99	-	321.00	321.00	0	0.990744	1.0
8	0.95	0.90	-	173.00	173.00	0	0.942755	1.0
8	0.95	0.95	179*	184.00	184.00	0	0.974181	1.0
8	0.95	0.99	-	247.00	247.00	0	0.991377	1.0
9	0.90	0.90	239*	239.00	239.00	0	0.906564	1.0
9	0.90	0.95	286*	286.00	287.50	15	0.956670	1.0
9	0.90	0.99	-	401.00	401.80	1	0.990750	1.0
9	0.95	0.90	-	204.00	204.50	5	0.928789	1.0
9	0.95	0.95	209*	209.00	209.00	0	0.966935	1.0
9	0.95	0.99	-	286.00	289.00	15	0.990752	1.0
10	0.90	0.90	154*	154.00	154.00	0	0.905014	1.0
10	0.90	0.95	197*	197.00	197.20	2	0.951644	1.0
10	0.90	0.99	-	283.00	283.40	1	0.990793	1.0
10	0.95	0.90	-	136.00	136.00	0	0.961130	1.0
10	0.95	0.95	136*	136.00	136.00	0	0.961130	1.0
10	0.95	0.99	-	206.00	207.80	3	0.990621	1.0
15	0.90	0.90	-	225.00	227.00	4	0.901397	1.0
15	0.90	0.95	317	262.00	263.80	6	0.953308	1.0
15	0.90	0.99	-	373.00	377.00	11	0.990589	0.984
15	0.95	0.90	-	170.00	178.00	29	0.913171	1.0
15	0.95	0.95	-	196.00	198.70	12	0.950277	1.0
15	0.95	0.99	-	262.00	264.70	11	0.990486	1.0
20	0.90	0.90	-	192.00	209.80	40	0.900449	1.0
20	0.90	0.95	-	249.00	257.10	21	0.951573	0.964
20	0.90	0.99	-	357.00	373.60	30	0.990564	0.954
20	0.95	0.90	-	147.00	154.20	16	0.911795	1.0
20	0.95	0.95	926	160.00	165.30	18	0.950021	1.0
20	0.95	0.99	-	248.00	254.80	17	0.990022	1.0
25	0.90	0.90	-	322.00	338.00	41	0.901312	1.0
25	0.90	0.95	-	391.00	405.40	32	0.950441	1.0
25	0.90	0.99	-	518.00	533.10	28	0.991245	0.9998
25	0.95	0.90	1606	247.00	258.60	31	0.903600	1.0
25	0.95	0.95	-	271.00	277.90	20	0.951735	1.0
25	0.95	0.99	-	390.00	407.90	43	0.990752	1.0

* The optimal solution reported in (Dengiz et al., 1997a).

Table 26.3 Computational results for the test problems with different arc reliabilities.

				Breakdown of the Computational Effort in Percent			
n	R_{min}	Solutions Search	CPU per Solution	Factoring	Simulation $K = 200$	Simulation $K = 20,000$	Upper Bound
10	0.900	23468.5	0.00267	36.08	44.85	0.00	19.07
10	0.950	20083.9	0.00267	23.70	41.74	0.00	34.55
10	0.990	24737.9	0.00284	7.97	39.08	0.00	52.94
19	0.900	30123.3	0.00498	3.44	96.54	0.01	0.00
19	0.950	32827.8	0.00501	14.04	85.93	0.02	0.00
19	0.990	34049.4	0.01729	37.08	50.20	4.92	7.80
19	0.995	37834.1	0.02836	13.24	44.59	9.13	33.04

Table 26.4 Results for the test problems with different arc reliabilities.

			Results of the Genetic Algorithm				
n	R_{min}	Best Solution Reported	Best Cost	Mean Cost	Range	Best Reliability	p-Value
10	0.900	-	3792.92	3868.18	166	0.902018	1.0
10	0.950	5661.32	4403.93	4560.89	318	0.950242	1.0
10	0.990	-	5843.50	5936.91	253	0.990014	1.0
19	0.900	-	1292390.00	1428689.60	291536	0.902125	1.0
19	0.950	-	1348283.00	1406466.00	262051	0.952263	1.0
19	0.990	7694708.00	1619322.00	1665539.90	160733	0.990015	0.969
19	0.995	-	1805600.00	1854732.50	103731	0.995006	1.0

where $\tau_{ij}(\mathbf{X}) = 1$ if nodes i and j are connected in state $\mathbf{X}$, $\tau_{ij}(\mathbf{X}) = 0$ if they are not, t_{ij} is the two-way traffic average demand between nodes i and j, and $\gamma = \sum_{i<j\leq n} t_{ij}$. By assigning proper values to t_{ij}, $T(G)$ can represent different network resilience measures. For example, if $t_{ij} = 1$ for each node pair i and j, $T(G)$ gives the expected fraction of node pairs communicating.

There are several important differences between network resilience and connectivity based reliability measures such as all-terminal reliability. First, all-terminal reliability implicitly ignores the effect of node failures in the reliability function since all nodes must be operational as the minimum requirement. The probability that all nodes are operational (i.e., the product of individual node reliabilities) can be incorporated into the reliability function as a constant term after which the reliability analysis is carried out without node failures (Colbourn, 1987). In other words, node failures have the same effect on all-terminal reliability independent of network topology. Therefore, the majority of the research on the reliable network design problem assumes perfectly reliable nodes. On the other hand, node failures must be taken into account while calculating a network resilience measure since a network's service availability in disconnect states is also of interest. While modeling telecommunication networks as probabilistic graphs, in fact, nodes represent complex processing units that are more likely to fail than highly reliable links (cables or microwaves) represented by arcs. Therefore, the assumption of perfectly reliable nodes does not represent reality in telecommunication networks.

Another drawback of all-terminal reliability is that disconnectivity of all nodes is considered equally in the reliability function, meaning that disconnectivity of a node with a large amount of incoming and outgoing traffic and a node with a small amount of traffic have equal weights in the reliability function. In both cases, the network service level is zero (i.e., the network is disconnected). Because of these reasons, network resilience measures are more versatile than pure connectivity based measures such as all-terminal reliability in modeling connectivity of telecommunication networks.

26.6.2 A Bi-objective Genetic Algorithm to Design Resilient Networks

In this section, the resilient network problem is formulated as a bi-objective problem with a 2-node connectivity constraint. Multiple conflicting objectives are common in most real-world telecommunication network design problems. In fact, the overall task of designing networks involves several phases, in which many conflicting objectives are considered. Choosing a network topology is the first step of the network design process. Candidate topologies are refined in the detailed design phases based on the specifications and available technology. Therefore, in topological design, cost and resilience (or reliability) are not hard constraints but competing objectives. The objective in this section is to introduce an approach to aid network designers in choosing a final network design by providing the trade-off curve of cost and network resilience in terms of a set of network topologies that are not inferior to each other with respect to cost or resilience.

26.6.2.1 Bi-objective optimization. If the objectives under consideration conflict among each other, optimizing with respect to a single objective often results in unacceptable results with respect to the other objectives. Therefore, a perfect multi-objective solution that simultaneously optimizes each objective function is almost impossible. A reasonable solution to a multi-objective problem is to investigate a set of solutions, each of which satisfies the objectives at an acceptable level without being dominated by any other solution. The ultimate goal of a multi-objective optimization is to identify the Pareto optimal set, which is the set of feasible solutions which are not dominated by any other feasible solutions in the solution space.

In our case, a solution X is said to dominate another solution Y if and only if at least one of the following conditions is satisfied.

(i) $C(X) \leq C(Y)$ and $T(X) > T(Y)$
(ii) $C(X) < C(Y)$ and $T(X) \geq T(Y)$.

A feasible solution X is said to be Pareto optimal if it is not dominated by any other feasible solution Y in the solution space. A true Pareto optimal solution cannot be improved with respect to any objective without worsening at least one other objective.

The goal of a multi-objective algorithm is to find the set of all non-dominated solutions in the solution space, called the Pareto set. The representation of Pareto set in the objective space is called Pareto front. Depending on the problem, the Pareto set may be very large, making it very expensive or impossible to investigate fully. In many cases, therefore, investigating a set of solutions uniformly approximating the true Pareto front is preferable.

There are alternative approaches to multi-objective optimization such as, weighted sums of objectives, alternating objectives, and Pareto ranking. In the recent decade, GA has become a popular heuristic tool for multi-objective optimization problems. This popularity can be attributed to its multi-solution approach and its capability to exploit the similarities of Pareto optimal solutions in order to generate new solutions by means of crossover. Interested readers may refer to the comprehensive survey papers by Fonseca and Fleming (1995), Deb (1999), and Van Veldhuizen and Lamont (2000).

26.6.2.2 Overall Optimization Algorithm. The bi-objective GA uses the same problem encoding scheme, crossover operator, and mutation operators as the single objective GA given in the previous section. Generally, a multi-objective GA has two sets of solutions, the population where general GA operations are applied and the elitist list storing all non-dominated solutions found so far during the search. However, the bi-objective GA does not use an elitist list and its population is made of only non-dominated solutions found so far in the search.

The overall procedure of the bi-objective GA is given below. In each iteration, a single solution is generated by either crossover or mutation, which is selected randomly and uniformly. In mutation, one of the mutation operators is also randomly and uniformly selected. Iterations continue until a maximum number of new solutions (g_{max}) is reached. The overall procedure of the bi-objective GA is as follows:

Step 1. Generate μ_0 nondominated initial solutions. $t = 1$.

Step 2. **If** $U \leq 0.5$, **Then** go to Step 3 **Else** Step 4.
Step 3. Crossover:

Step 3.1 Randomly select two solutions X and Y, and crossover them to produce new solution Z.

Step 3.2 Evaluate solution Z, and update the population if necessary. Go to Step 5.

Step 4. Mutation:

Step 4.1 Randomly select a solution X, and randomly and uniformly choose a mutation operator, and mutate solution X to generate new solution Z.

Step 4.2 Evaluate solution Z, and update the population if necessary. Go to Step 5.

Step 5. Stop if $t \geq g_{max}$.
Step 6. **If** $mod(t, K_{step}) = 0$, **Then** perform K additional replications for each solution in the population and update estimations using (26.13) and (26.14). Check the population one more time with the new estimates and remove any dominated solutions.
Step 7. Set $t = t + 1$. Go to Step 3.

26.6.2.3 Calculation of Traffic Efficiency and Evaluation of Solutions. The exact calculation of traffic efficiency, which requires examining 2^{n+m} network states, is intractable. In a sense, the exact calculation of $T(G)$ is more difficult than the exact calculation of $R(G)$ since node failures must be considered. In addition, enumeration schemes such as factoring (Page and Perry, 1991) and domination theory (Satyanarayana and Chang, 1983), which exploit the binary nature of the reliability structure function to improve performance of enumeration, cannot be used with the same efficiency. The most probable state bounds are the only bounds applicable to estimate $T(G)$. However, these bounds were very inconsistent in our initial experiments, meaning that the upper bound did not correlate well with the actual $T(G)$. Therefore, a simulation based on Sequential Construction Sampling (Easton and Wong, 1980) is used to estimate $T(G)$. The details of the simulation procedure will not be included herein.

Estimating traffic efficiency is a very computationally intense operation. To reduce the computational effort at the beginning of the search, $T(G)$ is estimated using a low number of simulation replications assuming that solutions at the beginning are inferior. As the search progresses, the number of replications is gradually increased to the maximum replication number K_{max}, which is done in K_{step} steps by performing K_{max}/K_{step} additional replications for each solution in the population in every new g_{max}/K_{step} solutions evaluated.

This process requires updating the estimate for each solution in the current population, which is performed as follows. Let $T(Z, K_1)$ and $\sigma^2_{T(Z,K_1)}$ be estimated traffic efficiency and the variance of the estimation, respectively, using total K_1 replications. After K_2 additional replications, the new estimate can calculated as follows:

$$T(Z, K_1 + K_2) = \frac{K_1 \times T(Z, K_1) + K_2 \times T(Z, K_2)}{K_1 + K_2} \tag{26.13}$$

Table 26.5 The coordinates of the nodes for the 10- and 20-node problems

node	1	2	3	4	5	6	7	8	9	10
x	26	38	93	74	86	60	26	44	54	52
y	5	86	64	8	61	10	70	70	71	36

node	11	12	13	14	15	16	17	18	19	20
x	57	13	32	93	7	33	54	49	99	50
y	28	68	9	56	42	52	13	3	56	27

with the following variance

$$\sigma^2_{T(Z,K_1+K_2)} = \frac{K_1^2 \times \sigma^2_{T(Z,K_1)} + K_2^2 \times \sigma^2_{T(Z,K_2)}}{(K_1+K_2)^2} \tag{26.14}$$

26.6.2.4 Computational Experiments. A 10-node and a 20-node problem were used to demonstrate the effectiveness of the bi-objective GA. The x and y coordinates of the nodes for both problems are given in Table 26.5. The 10-node problem uses the first 10 nodes of the 20-node problem. The cost of each arc is equal to the Euclidean distance between its two-end nodes. For both problems, all arcs and nodes have reliabilities of .97 and .99, respectively. The parameters of the bi-objective GA in all runs were as follows: $g_{max} = 100{,}000$, and for simulation, $K_{max} = 40{,}000$ and $K_{step} = 10$.

Figure 26.1 shows the Pareto front found for the 20-node problem. As seen in the figure, the final Pareto front is diverse, and it significantly improved upon the initial one. A similar result was obtained for the 10-node problem. Figure 26.2 illustrates the network designs found at the opposing ends and in the middle of the Pareto front for the 20-node problem. The least resilient and cheapest solution is made of two cycles; a similar cycle based structure was also observed at the low resilience end of the Pareto front for the 10-node problem. The most resilient and expensive solution found is not fully dense although in theory the fully dense network is the most resilient network. However, after a level of density, adding arcs improves resilience only at a negligible level since node failures becomes main reason for disconnectivity. Therefore, at the high-cost/high-resilience end of the Pareto front, solutions did not reach full-density. Another interesting observation is that the nodes have almost a uniform number of links (3 or 4). When the nodes are subject to failure, over connecting a node really does not really improve resilience. These examples provide convincing evidence that the bi-objective GA is capable of identifying a wide spectrum of solutions for large problems in a single run.

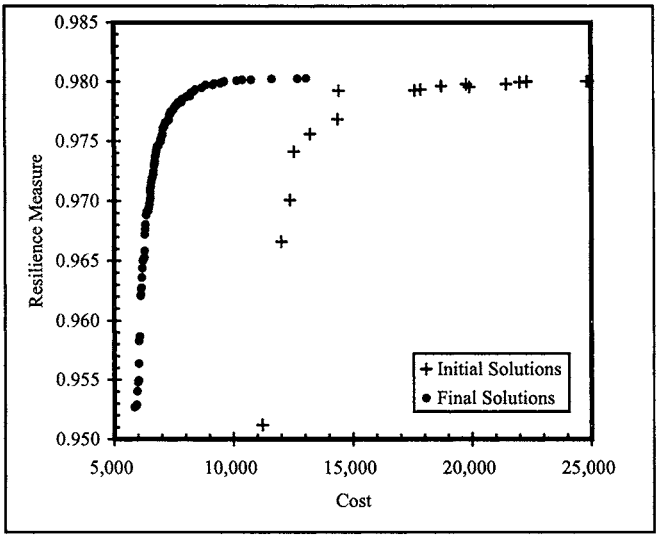
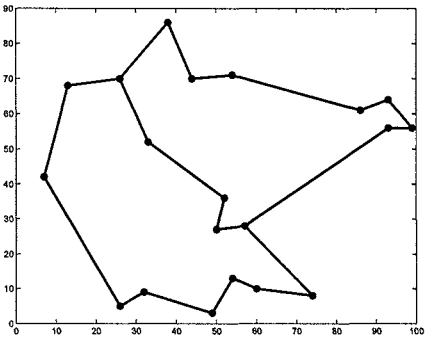


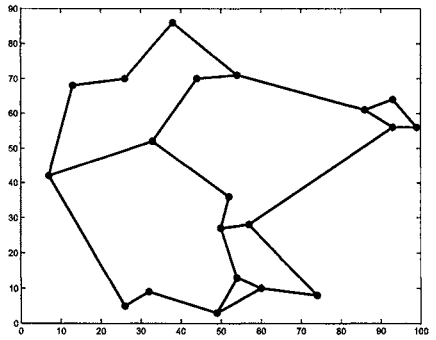
Figure 26.1 The Pareto Front found for the 20-node problem.

26.7 CONCLUSIONS

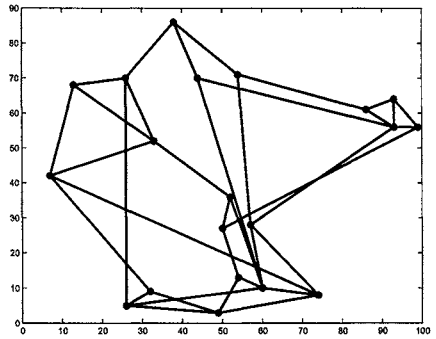
Network reliability has become important criteria because of recent technological innovations that encourage sparse networks and high traffic. Computing and estimating a network reliability measure is a difficult task. This chapter presented two GAs to solve the reliable and resilient network design problem considering a survivability constraint. The main focus of the algorithms was to efficiently evaluate the reliability of candidate solutions. State-of-the-art techniques are introduced and used to evaluate reliability, making it possible to solve large problems in relatively short time, even by exactly computing reliability.



(a) The least resilient solution



(b) A solution between the most and least resilient solutions



(c) The most resilient solution

Figure 26.2 Sample solutions found for the 20-node problem.

H.M. Aboelfotoh and C.J. Colbourn. Series-parallel bounds for the two-terminal reliability problem. *ORSA Journal on Computing*, 1(4):209–22, 1989.

H.M.F. AboElFotoh and L.S. Al-Sumait. A neural approach to topological optimization of communication networks, with reliability constraints. *IEEE Transactions on Reliability*, 50(4):397–408, 2001.

K.K. Aggarwal, Y.C. Chopra, and J.S. Bajwa. Topological layout of links for optimising the overall reliability in a computer communication system. *Microelectronics and Reliability*, 22(3):347–51, 1982a.

K.K. Aggarwal, Y.C. Chopra, and J.S. Bajwa. Topological layout of links for optimizing the s-t reliability in a computer communication system. *Microelectronics and Reliability*, 22(3):341–5, 1982b.

A. Balakrishnan, T.L. Magnanti, and P. Mirchandani. Designing hierarchical survivable networks. *Operations Research*, 46(1):116–36, 1998.

M.O. Ball. Computing network reliability. *Operations Research*, 27:823–38, 1979.

M.O. Ball. Complexity of network reliability computations. *Networks*, 10(2):153–65, 1980.

M.O. Ball, C.J. Colbourn, and J.S. Provan. *Network reliability*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 673–672. Elsevier Science B.V., Amsterdam, 1995.

M.O. Ball and G.L. Nemhauser. Matroids and a reliability analysis problem. *Mathematics of Operations Research*, 4(2):132–43, 1979.

M.O. Ball and J.S. Provan. Calculating bounds on reachability and connectedness in stochastic networks. *Networks*, 13(2):253–78, 1983.

M.O. Ball and R. Van Slyke. Backtracking algorithms for network reliability analysis. *Discrete Applied Mathematics*, 1:49–64, 1977.

S.G. Belovich. A design technique for reliable networks under a nonuniform traffic distribution. *IEEE Transactions on Reliability*, 44(3):377–87, 1995.

T.B. Brecht and C.J. Colbourn. Lower bounds on two-terminal network reliability. *Discrete Applied Mathematics*, 21(3):185–98, 1988.

D.W. Coit and A.E. Smith. Penalty guided genetic search for reliability design optimization. *Computers and Industrial Engineering*, 30(4):895–904, 1996.

D.W. Coit, A.E. Smith, and D.M. Tate. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing*, 8(2):173–82, 1996.

C.J. Colbourn. *Combinatorics of network reliability*. Oxford, New York, NY, USA, 1987.

D.L. Deeter and A.E. Smith. Economic design of reliable networks. *IIE Transactions*, 30(12):1161–74, 1998.

J. deMercado, N. Spyratos, and B.A. Bowen. A method for calculation of network reliability. *IEEE Transactions on Reliability*, R25(2):71–6, 1976.

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

B. Dengiz, F. Altiparmak, and A.E. Smith. Local search genetic algorithm for optimal design of reliable networks. *IEEE Transactions on Evolutionary Computation*, 1 (3):179–88, 1997b.

M.C. Easton and C.K. Wong. Sequential destruction method for monte carlo evaluation of system reliability. *IEEE Transactions on Reliability*, R-29(1):27–32, 1980.

T. Elperin, I. Gertsbakh, and M. Lomonosov. Estimation of network reliability using graph evolution models. *IEEE Transactions on Reliability*, 40(5):572–81, 1991.

G.S. Fishman. A comparison of four monte carlo methods for estimating the probability of s-t connectedness. *IEEE Transactions on Reliability*, R-35(2):145–55, 1986a.

G.S. Fishman. A monte carlo sampling plan for estimating network reliability. *Operations Research*, 34(4):581–94, 1986b.

J.N. Hagstrom. Using the decomposition tree for directed-network reliability computation. *IEEE Transactions on Reliability*, R-33(5):390–5, 1984.

Rong-Hong Jan. Design of reliable networks. *Computers and Operations Research*, 20(1):25–34, 1993.

Rong-Hong Jan, Fung-Jen Hwang, and Sheng-Tzong Chen. Topological optimization of a communication network subject to a reliability constraint. *IEEE Transactions on Reliability*, 42(1):63–70, 1993.

H. Kumamoto, K. Tanaka, and K. Inoue. Efficient evaluation of system reliability by monte carlo method. *IEEE Transactions on Reliability*, R-26(5):311–15, 1977.

H. Kumamoto, K. Tanaka, K. Inoue, and E.J. Henley. Dagger-sampling monte carlo for system unavailability evaluation. *IEEE Transactions on Reliability*, R-29(2): 122–5, 1980.

A. Kumar, R.M. Pathak, and Y.P. Gupta. Genetic-algorithm-based reliability optimization for computer network expansion. *IEEE Transactions on Reliability*, 44 (1):63–72, 1995a.

A. Kumar, R.M. Pathak, Y.P. Gupta, and H.R. Parsaei. A genetic algorithm for distributed system topology design. *Computers and Industrial Engineering*, 28(3): 659–70, 1995b.

Y.F. Lam and V.O.K. Li. An improved algorithm for performance analysis of networks with unreliable components. *IEEE Transactions on Communications*, COM-34(5): 496–7, 1986.

V.O.K. Li and J.A. Silvester. Performance analysis of networks with unreliable components. *IEEE Transactions on Communications*, COM-32(10):1105–10, 1984.

M.V. Lomonosov and V.P. Polesskii. Lower bound of network reliability. *Problemy Peredachi Informatsii*, 8(2):118–23, 1972.

M. Mazumdar, D.W. Coit, and K. McBride. A highly efficient monte carlo method for assessment of system reliability based on a markov model. *American Journal of Mathematical and Management Sciences*, 19(1-2):115–33, 1999.

C.L. Monma and D.F. Shallcross. Methods for designing communications networks with certain two-connected survivability constraints. *Operations Research*, 37(4): 531–41, 1989.

L.B. Page and J.E. Perry. A note on computing environments and network reliability. *Microelectronics and Reliability*, 31(1):185–6, 1991.

J.S. Provan and M.O. Ball. Computing network reliability in time polynomial in the number of cuts. *Operations Research*, 32(3):516–26, 1984.

L.I.P. Resende. Implementation of a factoring algorithm for reliability evaluation of undirected networks. *IEEE Transactions on Reliability*, 37(5):462–8, 1988.

M.G.C. Resende. A program for reliability evaluation of undirected networks via polygon-to-chain reductions. *IEEE Transactions on Reliability*, R-35(1):24–9, 1986.

B. Sanso, M. Gendreau, and F. Soumis. An algorithm for network dimensioning under reliability considerations. *Annals of Operations Research*, 36(1-4):263–74, 1992.

A. Satyanarayana and M.K. Chang. Network reliability and the factoring theorem. *Networks*, 13(1):107–20, 1983.

A. Satyanarayana and A. Prabhakar. New topological formula and rapid algorithm for reliability analysis of complex networks. *IEEE Transactions on Reliability*, R27(2): 82–100, 1978.

B. Singh and S.K. Ghosh. Network reliability evaluation by decomposition. *Microelectronics and Reliability*, 34(5):925–7, 1994.

R. Van Slyke and H. Frank. Network reliability analysis. i. *Networks*, 1(3):279–90, 1972.

C. Srivaree-ratana, A. Konak, and A.E. Smith. Estimation of all-terminal network reliability using an artificial neural network. *Computers and Operations Research*, 29(7):849–68, 2002.

C.-L. Yang and P. Kubat. Efficient computation of most probably states for communication networks with multimode components. *IEEE Transactions on Communications*, 37(5):535–8, 1989.