Evolutionary Methods for Design of Reliable Networks

a chapter in *Telecommunications Optimisation: Heuristic and Adaptive Methods* (D. Corne, editor), Wiley, forthcoming.

Alice E. Smith
Department of Industrial Engineering
University of Pittsburgh
1031 Benedum Hall
Pittsburgh, PA 15261 USA
aesmith@engrng.pitt.edu
www.pitt.edu/~aesmith

Berna Dengiz
Department of Industrial Engineering
Gazi University
06570 Maltepe, Ankara Turkey
berna@mikasa.mmf.gazi.edu.tr

1. Introduction to the Design Problem

The problem of how to design a network so that certain constraints are met and one or more objectives are optimized is relevant in many real world applications such as telecommunications [2, 25, 27, 45], computer networking [10, 36], water systems [39], and oil and gas lines [21]. This chapter focuses on design of minimum cost reliable communications networks when a set of nodes and their topology are given, along with a set of possible bi-directional arcs to connect them. A variety of approaches are cited, however the previous work of the authors using genetic algorithms is discussed in detail. It must be noted that the design problem solved by these methods is a significant simplification of the real design problem. Telecommunications networks involve a large number of components and considerations that are not treated here. Instead, the approaches discussed in this chapter focus on the costs and reliabilities of the arcs (links) of the network.

1.1 Costs

Costs can include material costs of the cabling, installation costs such as trenching or boring, land or right of way costs, and connection or terminal costs inherent with the cabling. Many of these costs are "unit costs," that is, they depend on the length of the arc. However, there can be fixed costs per arc and these are easily accommodated in the methods discussed. In many papers, a unit cost is not specifically mentioned; instead each arc is assigned a weight which is used as the complete cost of the arc [3, 4, 32].

1.2 Reliability

Associated with each type of connection is a reliability (with an implicit mission time), or equivalently, a stationary availability. This reliability has a range from 0 (never operational) to 1 (perfectly reliable). It is assumed (with good justification) that reliability comes at a cost. Therefore, a more reliable connection type implies a greater unit cost. The trade-off between cost and reliability is not linear. An increase in reliability causes a greater than equivalent increase in cost; often a quadratic relationship is assumed. Other simplifying assumptions commonly made are that nodes are perfectly reliable and do not fail, and that arcs have two possible states - good or failed. Arcs fail independently and repair is not considered.

There are two main reliability measures used in network design, namely all-terminal[1] and source-sink[2]. Sections 4 and 5 of this chapter consider only all-terminal reliability while section 6 includes a source-sink reliability problem. All-terminal network reliability is concerned with the ability of each and every network node to be able to communicate with all other network nodes through some (non-specified) path. This implies that the network form at least a minimum spanning tree. Source-sink reliability is concerned with the ability of the source node (pre-

[1] This is also called *uniform* or *overall* network reliability.
[2] This is also called *two terminal* network reliability.

specified) to communicate with the sink node (also pre-specified) through some (non-specified) path.

The problem of calculating or estimating the reliability of a network is an active area of research related to the network design problem. There are four main approaches — exact calculation through analytic methods, estimation through variations of Monte Carlo simulation, upper or lower bounds on reliability, and easily calculated, but crude, surrogates for reliability. The issue of calculating or estimating the reliability of the network is so important for optimal network design, section 3 covers it in detail.

1.3 Design Objectives and Constraints

The most common objective is to design a network by selecting a subset of the possible arcs so that network reliability is maximized and a maximum cost constraint is met. However, in many situations, it makes more sense to minimize network cost subject to a minimum network reliability constraint. There may be side constraints, such as minimum node degree[3] or maximum arc length allowed in the network. In this chapter, the objective is to find the minimum cost network architecture that meets a pre-specified minimum network reliability:

$$\text{Min: } C(\mathbf{x})$$

$$\text{s.t. } R(\mathbf{x}) \geq R_o$$

1.4 Difficulty of the Problem

The network design problem, as described, is an NP-hard combinatorial optimization problem [19] where the search space for a fully connected network with *N* nodes and *k* possible arc choices is:

[3] Node degree is simply the number of arcs emanating from the node.

$$k^{\frac{(|N|\times(|N|-1))}{2}} \quad (1)$$

Compounding the exponential growth in number of possible network topologies is that the exact calculation of network reliability is also an NP-hard problem, which grows exponentially with the number of arcs.

1.5 Notation Used in the Rest of the Chapter:

N Set of given nodes.

L Set of possible arcs.

l_{ij} Option of each arc. $l_{ij} \in \{1,2,..., k\}$

$p(l_k)$ Reliability of arc option.

$c(l_k)$ Unit cost of arc option.

x Topology of a network design.

C(**x**) Total cost of a network design.

C_o Maximum cost constraint.

R(**x**) Reliability of a network design.

R_o Minimum network reliability constraint.

g GA generation.

s GA population size per generation.

m% GA percentage of mutants created each generation.

r_p GA penalty rate.

r_m GA mutation rate.

t Number of Monte Carlo reliability simulation iterations.

2. A Sampling of Optimization Approaches

The optimal design problem when considering reliability has been studied in the literature

using alternative methods of search and optimization. Jan et al. [25] developed an algorithm using decomposition based on branch and bound to minimize arc costs with a minimum network reliability constraint; this is computationally tractable for fully connected networks up to 12 nodes. Using a greedy heuristic, Aggarwal et al. [3] maximized reliability given a cost constraint for networks with differing arc reliabilities and an all-terminal reliability metric. Ventetsanopoulos and Singh [44] used a two-step heuristic procedure for the problem of minimizing a network's cost subject to a reliability constraint. The algorithm first used a heuristic to develop an initial feasible network configuration, then a branch and bound approach was used to improve this configuration. A deterministic version of simulated annealing was used by Atiqullah and Rao [4] to find the optimal design of very small networks (five nodes or less). Pierre et al. [36] also used simulated annealing to find optimal designs for packet switch networks where delay and capacity were considered, but reliability was not. Tabu search was used by Glover et al. [20] to choose network design when considering cost and capacity, but not reliability. Another tabu search approach by Beltran and Skorin-Kapov [7] was used to design reliable networks by searching for the least cost spanning 2-tree, where the 2-tree objective was a crude surrogate for reliability. Koh and Lee [27] also used tabu search to find telecommunication network designs that required some nodes (special offices) have more than one arc while others (regular offices) required only one arc, using this arc constraint as a surrogate for network reliability.

Genetic algorithms (GA) have recently been used in combinatorial optimization approaches to reliable design, mainly for series and parallel systems [11, 23, 35]. For network design, Kumar et al. [32] developed a GA considering diameter, average distance, and computer network reliability and applied it to four test problems of up to nine nodes. They calculated all-

terminal network reliability exactly and used a maximum network diameter (minimal number of arcs between any two nodes) as a constraint. The same authors used this GA to design the expansion of existing computer networks [33]. Their GA approach has two significant limitations. First, they require that all network designs considered throughout the search be feasible. While this is relatively easy to achieve using a cost constraint and a maximum reliability objective, it is not as easy when using a cost objective and a reliability constraint. The second limitation is their encoding, which is a list of all possible arcs from each node, arranged in an arbitrary node sequence. The presence of an arc is signaled by a 1 and its absence by a 0. For a ten node problem, the encoding grows to a string length of 90. However, the more serious drawback of the encoding is the difficulty in maintaining the agreement of the arcs present and absent after crossover and mutation. An elaborate repair operator must be used, which tends to disrupt the beneficial effects of crossover. Davis et al. [13] approached a related problem considering arc capacities and re-routing upon arc failure using a problem-specific GA. Abuali et al. [1] assigned terminal nodes to concentrator sites to minimize costs while considering capacities using a GA, but no reliability was considered. The same authors [2] solved the probabilistic minimum spanning tree problem where inclusion of the node in the network is stochastic and the objective is to minimize connection (arc) costs, again without regard to reliability. Walters and Smith [45] used a GA to address optimal design of a pipe network that connects all nodes to a root node using a non-linear cost function. Reliability and capacity were not considered.

3. The Network Reliability Calculation During Optimal Design

Iterative (improvement) optimization techniques depend on the calculation or estimation of the reliability of different network topologies throughout the search process. However, the

calculation of all-terminal network reliability is itself an NP-hard problem [37][4]. Assuming that the arcs (set L) fail independently, the number of the possible network states is 2^L. For large L, it is computationally impossible to calculate the exact network reliability using state enumeration even once, much less the numerous times required by iterative search techniques. Therefore, the main interest is in crude surrogates, simulation methods and bounding methods. Crude surrogates to network reliability include a constraint on minimum node degree or minimum path connectedness. These are easily calculated, however they are not precisely correlated with actual network reliability. For the all-terminal network reliability problem, efficient Monte Carlo simulation is difficult because simulation generally loses efficiency as a network approaches a fully connected state.

When considering bounds, both the tightness of the bound and its computational effort must be considered. Upper and lower bounds based on formulations from Kruskal [30] and Katona [26], as comprehensively discussed in [8], are based on the reliability polynomial and can be used for both source-sink and all-terminal network reliability. The importance of the reliability polynomial is that it transforms the reliability calculation into a counting of operational network states on a reduced set of arcs. Bounds on the coefficients lead directly to bounds on the reliability polynomial. The accuracy of the Kruskal-Katona bounds depends on both the number and the accuracy of the coefficients computed. Ball and Provan [6] report tighter bounds by using a different reliability polynomial. Their bounds can be computed in polynomial (in L) time and are applicable for both source-sink and all-terminal reliability. Brecht and Colbourn [8] improved the Kruskal-Katona bounds by efficiently computing two additional coefficients of the polynomial. Brown et al. [9] used network transformations to

[4] Much of this discussion also applies to source-sink reliability, which is also NP-hard but easier than all-terminal network reliability.

efficiently compute the Ball-Provan bounds for all-terminal reliability. Nel and Colbourn [34] developed a Monte Carlo method for estimating some additional coefficients in the reliability polynomial of Ball and Provan. These additional coefficients provide substantial improvements (i.e., tighter bounds). Another efficiently computable all-terminal reliability upper bound is defined by Jan [24]. Jan's method uses only the cut sets separating individual nodes from a network and can be calculated in polynomial (in N) time. Note the distinction between polynomial in N (nodes) and polynomial in L (arcs), where for highly reliable networks, L will far exceed N.

One of the important limitations of the bounding methods cited is that they require that all arcs have the same reliability, which is an unrealistic assumption for many problems. In new work by Konak and Smith [28, 29], Jan's approach [24] is extended to networks with unequal arc reliability. Also, a tighter upper bound is achieved, even for the case when all arc reliabilities are identical, at virtually no additional computational cost, i.e., the new bound is polynomial in N.

For solving the optimal design problem, it is likely that a combination of crude surrogates, bounding the network reliability along with accurately estimating it with Monte Carlo simulation will be a good approach. Through much of the search, crude surrogates or bounds will be accurate enough, however as the final few candidate topologies are weighed, a very accurate method must be employed (simulation with many replications or exact calculation).

4. A Simple Genetic Algorithm Method When All Arcs have Identical Reliability

In this section, a simple GA approach to optimal network design when all arcs have identical reliability is discussed. This approach was developed by Dengiz et al. [16].

4.1 Encoding and GA Operators

Each candidate network design is encoded as a binary string with the size of N (N-1)/2, the number of possible arcs in a fully connected network.[5] For example, Figure 1 shows a simple network that consists of 5 nodes and 10 possible arcs, but with only 7 arcs present.

Figure 1 here.

The string representation of this network is:

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} x_{12} & x_{13} & x_{14} & x_{15} & x_{23} & x_{24} & x_{25} & x_{34} & x_{35} & x_{45} \end{bmatrix}$$

In this GA, the initial population consists of randomly generated 2-connected networks [38]. The 2-connectivity measure is used as a preliminary screening, since it is usually a property of highly reliable networks. A set of experiments determined the following GA parameter values: s = 20, r_c = 0.95, and r_m = 0.05.

The approach uses the conventional GA operators of roulette wheel selection, single point crossover and bit flip mutation. Below is an example of the single point crossover (with the splice point after x_{15}) and bit flip mutation [21]. Evolution continues until a preset number of generations, which varies according to the size of the network, N.

Parent 1

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} x_{12} & x_{13} & x_{14} & x_{15} & x_{23} & x_{24} & x_{25} & x_{34} & x_{35} & x_{45} \end{bmatrix}$$

Parent 2

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$
$$\begin{bmatrix} x_{12} & x_{13} & x_{14} & x_{15} & x_{23} & x_{24} & x_{25} & x_{34} & x_{35} & x_{45} \end{bmatrix}$$

[5] This is reduced for networks where not all possible links are permitted, as demonstrated on test problems 18 - 20.

Child 1

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$
$$\begin{bmatrix} x_{12} & x_{13} & x_{14} & x_{15} & x_{23} & x_{24} & x_{25} & x_{34} & x_{35} & x_{45} \end{bmatrix}$$

Mutated Child 1

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$
$$\begin{bmatrix} x_{12} & x_{13} & x_{14} & x_{15} & x_{23} & x_{24} & x_{25} & x_{34} & x_{35} & x_{45} \end{bmatrix}$$

Child 2

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} x_{12} & x_{13} & x_{14} & x_{15} & x_{23} & x_{24} & x_{25} & x_{34} & x_{35} & x_{45} \end{bmatrix}$$

Mutated Child 2

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$
$$\begin{bmatrix} x_{12} & x_{13} & x_{14} & x_{15} & x_{23} & x_{24} & x_{25} & x_{34} & x_{35} & x_{45} \end{bmatrix}$$

4.2 The Fitness Function

The objective function is the sum of the total cost for all arcs plus a quadratic penalty function for networks that fail to meet the minimum reliability requirement. The objective of the penalty function is to lead the GA to near-optimal feasible solutions. It is important to allow infeasible solutions into the population because good solutions are often the result of breeding between a feasible and an infeasible solution and the GA does not ensure feasible children, even if both parents are feasible [12, 40]. The fitness function considering possible infeasible solutions is:

$$Z(\mathbf{x}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij}\, x_{ij} + \delta\, (c_{MAX}(R(\mathbf{x})-R_o))^2 \qquad (2)$$

where $\delta = 1$ if the network is infeasible and 0 otherwise. C_{MAX} is the maximum arc cost possible in the network.

4.3 Dealing with the Reliability Calculation

This method uses three reliability estimations to tradeoff accuracy with computational effort.

- A connectivity check for a spanning tree is made on all new network designs using the method of [22].
- For networks that pass this check, the 2-connectivity measure [38] is made by counting the node degrees.
- For networks that pass both of these preliminary checks, Jan's upper bound [24] is used to compute the upper bound of reliability of a candidate network, $R_U(\mathbf{x})$.

This upper bound is used in the calculation of the objective function (2) for all networks except those which are the best found so far ($\mathbf{x}_{BEST}$). Networks which have $R_U(\mathbf{x}) \geq R_o$ and the lowest cost so far are sent to the Monte Carlo subroutine for more precise estimation of network reliability using an efficient Monte Carlo technique by Yeh et al. [46]. The simulation is done for t=3000 iterations for each candidate network.

4.4 Computational Experiences

Results compared to the branch-and-bound method of Jan et al. [25] on the test problems are summarized in Table 1. These problems are both fully connected and non-fully connected networks (*viz.*, only a subset of L is possible for selection). N of the networks ranges from 5 to 25. Each problem for the GA was run 10 times, each time with a different random number seed. As shown, the GA gives the optimal value for the all replications of problems 1 - 3 and finds optimal for all but two of the problems for at least one run of the 10. The two with suboptimal results (12 and 13) are very close to optimal. Table 2 lists the search space for each problem along with the proportion actually searched by the GA during a single run ($n \times g_{MAX}$). g_{MAX}

ranged from 30 to 20000, depending on problem size. This proportion is an upper bound because GA's can (and often do) revisit solutions already considered earlier in the evolutionary search. It can be seen that the GA approach examines only a very tiny fraction of the possible solutions for the larger problems, yet still yields optimal or near-optimal solutions. Table 2 also compares the efficacy of the Monte Carlo estimation of network reliability. The exact network reliability is calculated using a backtracking algorithm also used by Jan et al. [25] and compared to the estimated counterpart for the final network for those problems where the GA found optimal. The reliability estimation of the Monte Carlo method is unbiased and is always within 1% of the exact network reliability.

Tables 1 and 2 here.

5. A Problem-Specific Genetic Algorithm Method When All Arcs have Identical Reliability

The GA in the preceding section was effective, but there are greater computational efficiencies possible if the GA can exploit the particular structure of the optimal network design problem. This section presents such an approach as done in [17, 18]. The encoding, crossover and mutation are modified to perform local search and repair during evolution and the initial population is seeded. These modifications improve both the efficiency and the effectiveness of the search process. The drawback, of course, is the work and testing necessary to develop and implement effective operators and structures.

5.1 Encoding and Seeding

A variable-length integer string representation was used with every possible arc arbitrarily assigned an integer and the presence of that arc in the topology is shown by the presence of that integer in the ordered string. The fully connected network in figure 2 uses the following assignment:

Link	Integer Label
1,3	1
1,5	2
1,6	3
1,4	4
1,2	5
2,3	6
2,5	7
2,6	8
2,4	9
3,4	10
3,6	11
3,5	12
4,5	13
4,6	14
5,6	15

Insert Figure 2 here.

String representations of networks given in figure 2 are [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15] and [1 4 5 6 9 11 12 13 14 15], respectively. The first network includes all possible arcs using the labels above. The second network contains ten arcs, using the same labeling scheme.

The initial population consists of networks with the characteristics of being highly reliable. The initial population is generated by:

1. A spanning tree is implemented through the depth-first search algorithm by Hopcroft and Ullman [22], which grows a tree from a randomly chosen node.
2. Arcs selected randomly from the co-tree set (the set of arcs which are not yet used in the tree) are added to the spanning tree to increase connectivity.
3. If the network obtained by steps 1 and 2 does not have 2-connectivity [38], it is repaired by the algorithm explained in section 5.3.

5.2 The Genetic Algorithm

The flow of the algorithm is as follows:

Step 1: Generate the initial population. Calculate the fitness of each candidate network in the population using eq. 2 and Jan's upper bound [24] as R(**x**), except for the lowest cost network with $R_U(\mathbf{x}) \geq R_o$. For this network, $\mathbf{x}_{BEST}$, use the Monte Carlo estimation of R(**x**) in eq. 2. Generation, g, = 1.

Step 2: Select two candidate networks. An elitist ranking selection with stochastic remainder sampling without replacement is used [21].

Step 3: To obtain two children, apply crossover to the selected networks and to the children.

Step 4: Determine the 2-connectivity of each new child. Use the repair algorithm on any that do not satisfy 2-connectivity.

Step 5: Calculate $R_U(\mathbf{x})$ for each child using Jan's upper bound and compute its fitness using eq. 2.

Step 6: If the number of new children < s-1 go to Step 2.

Step 7: Replace parents with children, retaining the best solution from the previous generation.

Step 8: Sort the new generation according to fitness. i = 1 to s.

a) If $Z(\mathbf{x}_i) < Z(\mathbf{x}_{BEST})$, then calculate the reliability of this network using Monte Carlo simulation, else go to Step 9.

b) $\mathbf{x}_{BEST} = \mathbf{x}_i$. Go to Step 9.

Step 9 : If $g = g_{MAX}$ stop, else go to Step 2 and $g = g$+1.

The parameters of the GA are s = 50, r_c = 0.70, r_m = 0.30 and DR=0.60, a parameter used in mutation.

5.3 Repair, Crossover and Mutation

If any candidate network does not pass 2-connectivity, the network is repaired using three different alternatives according to how many nodes fail the test.

Step 1: Determine N_k, n_k; k=1,..., max node degree in a network.

Step 2: Rank all N_k and n_k, except $N_1$and n_1, in increasing order from k=2,.....,maximum node degree; determine N_{min} and n_{min}.

a) If n_1=1, determine which arc between this node and the nodes in the N_{min} set has minimum cost and add it, stop.

b) If n_1=2,

- Compute the connection cost of the 2 nodes ($c_{m_{11},m_{12}}$) in the N_1 set.

- Compute all $c_{m_{11},m_{\min j}}$ and $c_{m_{12},m_{\min j}}$ for j = 1,2,...,n_{min}.

- If $c_{m_{11},m_{12}} < [\min(c_{m_{11},m_{\min j}})+\min(c_{m_{12},m_{\min j}})]$ then connect the 2 nodes in the N_1 set; else connect the nodes in N_1 set to other nodes in N_{min}, through $\min(c_{m_{11},m_{\min j}})$, $\min(c_{m_{12},m_{\min j}})$.

c) If $n_1 > 2$,

- Randomly select 2 nodes from N_1 set,

- Apply (b) for these 2 nodes until $n_1 = 0$.

Where,

N_k	set of nodes with k degree
N_{min}	set of nodes with minimum degree except nodes with 1 degree
n_k	number of nodes in the N_k set
m_{1j}	node labels in the N_1 set
m_{minj}	node labels in the N_{min} set , j = 1,2,..., N_{min}

Crossover is a form of uniform crossover with repair to ensure each child is at least a spanning tree with 2-connectivity.

Step 1: Select two candidate networks, called T1 and T2. Determine the common arcs = T1$\cap$ T2, other arcs are: $\overline{T1}$ = T1 - (T1$\cap$ T2) ; $\overline{T2}$ = T2 - (T1$\cap$ T2)

Step 2: Assign common arcs to children, T1′, T2′. T1′ = T1∩T2; T2′ = T1∩T2

Step 3: If T1′ and T2′ are spanning trees, go to step 5, else go to step 4.

Step 4: Arcs from $\overline{T1}$, in cost order, are added to T1′ until T1′ is a spanning tree. Use the same procedure to obtain T2′ from $\overline{T2}$.

Step 5: Determine which arcs of T1 ∪ T2 do not exist in T1′ and T2′: CT1 = T1 \ T1′; CT2 = T2 \ T2′

Step 6: T1′ = T1′ ∪ CT2; T2′ = T2′ ∪ CT1

Mutation takes the form of a randomized greedy local search operator. The mutation operator is applied differently according to node degrees of the network.

Step 1: Determine node degrees deg(j) of the network for j = 1,2,...,N

If deg(j) = 2 for all j; go to Step 2,

If deg(j) > 2 for all j; go to Step 3,

Else, deg(j) ≥ 2; for all j; go to Step 4.

Step 2: Randomly select an allowable arc not in the network and add it; stop.

Step 3: Rank arcs of the network in decreasing cost order. Drop the maximum cost arc from the network. If the network still has 2-connectivity, stop; otherwise cancel dropping this arc, and retry the procedure for the remaining ranked arc until one is dropped or the list has been exhausted; stop.

Step 4: Generate u ~ U(0,1). If u <(1-DR) (where DR is the drop rate) go to step 2, otherwise go to step 3.

5.4 Computational Results

There are two comparisons made to judge the effectiveness and efficiency of this GA that uses local search in its crossover and mutation operators; these are the branch and bound (B+B)

technique by Jan et al. [25] and the GA of section 4. 79 randomly generated test problems are used for the comparison. Each problem for the GAs was run ten times with different random number seeds to gauge the variability of the GA.

Table 3 lists complete results of the three methods. The GAs find optimal solutions at a fraction of the computational cost of branch and bound for the larger problems. Both GA formulations found the optimal solution in at least one of the ten runs for all problems. The GA of this section is computationally more efficient and reduces variability among seeds when compared to the GA of section 4.

Insert Table 3 here.

6. A Genetic Algorithm When Links Can Have Different Reliabilities

A simplistic and restrictive assumption of previous research is that all possible arcs must have identical reliability and unit cost. This is a limitation of the approaches to the problem, not of the design problem itself. In real world design problems there are generally multiple choices for arcs, each with an associated reliability and unit cost, and other design attributes. When considering the economics of network design, it is important to allow designs with arcs of differing unit costs. The research presented in this section is from [14, 15] and makes the significant relaxation that there are multiple choices of arc type for each possible arc and the final network may have a heterogeneous combination of differing arc reliabilities and costs. This relaxation, while greatly improving the relevance of the problem to real world economic design, also complicates the network reliability calculation and exponentially increases the search space.

6.1 Encoding, Genetic Algorithm and Parameters

The following example for a problem with $N = 5$ and $k = 4$ levels of connection shows how a candidate network design is encoded as a chromosome. Notice that the same solution is

represented in figure 3. There are (5×4)/2 = 10 possible arcs for this example but only five are present; the other five are at level of connection $l_{i,j} = 0$. This information is placed in a chromosome using the possible values of 0,1,...*k*-1:

Chromosome: {**0100203102**}

Insert Figure 3 Here

The objective function of sections 4 and 5(eq. 2) is modified to:

$$C_p(\mathbf{x}) = C(\mathbf{x}) + C(\mathbf{x}^*) \times (1 + R_o - R(\mathbf{x}))^{r_p + \frac{s \times g}{50}} \qquad (3)$$

where $C_p(\mathbf{x})$ is the penalized cost, $C(\mathbf{x})$ is the unpenalized cost and $C(\mathbf{x}^*)$ is the cost of the best feasible solution in the population. This is a dynamic penalty that depends on the length of search, *g*.

Below is the GA algorithm, followed by a more detailed description of the key steps.

- **Randomly Generate Initial Population**
  - Send initial population to the reliability and cost calculation function and calculate fitness using eq. 3
  - Check for initial *Best Solution*
    - if no solution is feasible the best infeasible solution is recorded
- **Begin Generational Loop**
  - Select and Breed Parents
    - copy *Best Solution* to new population
    - two distinct parents are chosen using the rank based procedure of [43]
    - children are generated using uniform crossover
    - children are mutated

- when enough children are created the parents are replaced by the children

♦ Send new population to the reliability and cost calculation functions, and calculate fitness using eq. 3

♦ Check for new *Best Solution*

- if no solution is feasible the best infeasible solution is recorded

♦ Repeat until $g_{\max}$

Crossover is uniform by randomly taking an allele from one of the parents to form the corresponding allele of the child. This is done for each allele of the chromosome. For example, suppose parents $\mathbf{x_1}$ and $\mathbf{x_2}$ are chosen to breed.

$\mathbf{x_1}$ {0120131011}

$\mathbf{x_2}$ {1111012002}

_____

child {0110132001}

After a new child is created it goes through mutation. A solution undergoes mutation according to the percentage of population mutated. For example if m% = 20% and s = 30, then 6 members are randomly chosen and mutated. Once a solution is chosen to be mutated then the probability of mutation for each allele is equal to the mutation rate, r_m. Hence if r_m = 0.3 then each allele of the solution will be mutated with a 0.3 probability. When an allele is mutated its value *must* change. If an arc was turned off, $l_{i,j}$ = 0, then it will be turned on with an equal probability of being turned to any of the states 1 through k-1. If an allele is originally on, then it will either be turned off (k=0) or it will be turned to one of the different on levels, with equal probability. An example is seen below. The solution has been mutated by changing the seventh allele from a 2 to a 0 and changing the ninth allele from a 0 to a 1.

solution {0110132001}

mutated solution {0110130011}

6.2 Test Problem 1 - Ten Nodes

The ten node test problem was designed by randomly picking ten sets of (x,y) coordinates and using each of the points as nodes on an 100 by 100 grid. The Euclidean distances between the nodes were calculated and the unit costs and reliabilities were from table 4. The ten node problem was examined with a system reliability requirement of 0.95. Because of the network size, reliability could not be calculated exactly. The Monte Carlo estimator of reliability used both dynamic and static parameters. For the "general" reliability check, which was used on every new population member, the total number of replications used was dynamic. At the first generation, the estimator replicated each network 1,000 times (t = 1,000). As the number of generations increased, the number of replications used in the general reliability check also increased. After every hundredth generation the number of replications used in the general reliability check was incremented by 1000 ($t = t$ + 1,000). This dynamic approach was used so that as the search progressed the reliability estimates would get better. Whenever a network was created that met the reliability constraint using the general reliability estimator, and had a cost that was less than the best cost found so far, a "best check" reliability estimator was used. This replicated a given system t = 25,000 times. The best check was used to help ensure the feasibility and accuracy of the very best candidate designs.

From initial experimentation s = 90, m% = 25, r_c = 1.00, r_m = 0.25, r_p = 6, and $g_{\max}$ = 1,200. Since the problem has 1.24×10^{27} possible designs, it was impossible to enumerate. Thus a random greedy search was used to compare the effectiveness of the GA. Ten runs of each

algorithm using the same set of random number seeds were averaged and plotted as shown in figure 4.

Insert Figure 4 Here

Notice that the line corresponding to the best cost that the GA finds dips much more rapidly than does the best cost corresponding to the greedy algorithm indicating that the GA will find good solutions much more efficiently than a myopic approach. Also in the graph it is seen that both lines appear to be asymptotically approaching a solution, however the line corresponding to the GA is approaching a much better solution than the line corresponding to the greedy search.

6.3 Test Problem 2 - Source-Sink Reliability

This problem demonstrates the flexibility of the GA approach in two respects. First, the calculation of reliability is different. Second, the architecture of arcs is restricted; 18 of 36 arcs are unavailable for the network design as shown in figure 5. The GA easily accommodates these rather fundamental changes. The change in the reliability calculation is accomplished by simply modifying the backtracking algorithm of [5] (this problem is small enough to calculate reliability exactly during search). The fact that not all possible arcs are allowed in this design is accommodated by simply leaving these arcs out of the chromosome string as was done in some of the problems of sections 4 and 5.

Insert Figure 5 Here

This problem is taken from the literature [31] and has 6.9×10^{10} possible topologies, thus precluding enumeration to identify the optimal design. A system reliability requirement $R_o(\mathbf{x}) = 0.99$ is set. After some initial experimentation it was determined that $s = 40$, $m\% = 80$, $r_c = 1.00$, $r_m = 0.05$, $r_p = 6$, and $g_{\max} = 2{,}000$. Seven of the 10 runs found a best cost of 4680. The other

three test runs found a best cost of 4,726. Since the GA found only two distinct solutions over 10 runs, it is likely that both are near-optimal, if 4,680 is not optimal.

7. Concluding Remarks

It can be seen that an evolutionary approach to optimal network design when considering reliability is effective and flexible. Differences in objective function, constraints and reliability calculation are easily handled. One difficulty is the number of times that network reliability must be calculated or estimated. An effective search for problems of realistic size will use a combination of bounds, easily calculated reliability surrogates such as node degree, and Monte Carlo simulation estimators. Another emerging approach is to use artificial neural network approximators for network reliability [41, 42]. One important attribute of evolutionary search that has yet to be exploited in the literature is the generation of multiple, superior network designs during the procedure. A human designer could more carefully examine and consider the few superior designs identified by the evolutionary algorithm. These are likely to be dissimilar and thus show the designer the particularly promising regions of the design search space.

Acknowledgement

The authors gratefully acknowledge the support of U.S. National Science Foundation grant INT-9731207 and additional support from the Scientific and Technical Research Council of Turkey (Tubitak) for their collaboration.

References

[1] F. N. Abuali, D. A. Schoenefeld, R. L. Wainwright, "Terminal assignment in a communications network using genetic algorithms," *Proceedings of the ACM Computer Science Conference*, 1994, 74-81.

[2] F. N. Abuali, D. A. Schoenefeld, R. L. Wainwright, "Designing telecommunications networks using genetic algorithms and probabilistic minimum spanning trees," *Proceedings of the 1994 ACM Symposium on Applied Computing*, 1994, 242-246.

[3] K. K. Aggarwal, Y. C. Chopra, J. S. Bajwa, "Reliability evaluation by network

decomposition," *IEEE Transactions on Reliability*, vol. R-31, 1982, 355-358.

[4] M. M. Atiqullah, S. S. Rao, "Reliability optimization of communication networks using simulated annealing," *Microelectronics and Reliability*, vol. 33, 1993, 1303-1319.

[5] M. Ball, R. M. Van Slyke, "Backtracking algorithms for network reliability analysis," *Annals of Discrete Mathematics*, vol. 1, 1977, 49-64.

[6] M. O. Ball, J. S. Provan, "Calculating bounds on reachability and connectedness in stochastic networks," *Networks*, vol. 13, 1983, 253-278.

[7] H. F. Beltran, D. Skorin-Kapov, "On minimum cost isolated failure immune networks," *Telecommunications Systems*, vol. 3, 1994, 183-200.

[8] T. B. Brecht, C. J. Colbourn, "Lower bounds on two-terminal network reliability," *Discrete Applied Mathematics*, vol. 21, 1988, 185-198.

[9] J. I. Brown, C. J. Colbourn, J. S. Devitt, "Network transformations and bounding network reliability," *Networks*, vol. 23, 1993, 1-17.

[10] Y. C. Chopra, B. S. Sohi, R. K. Tiwari, K. K. Aggarwal, "Network topology for maximizing the terminal reliability in a computer communication network," *Microelectonics & Reliabiliy*, vol. 24, 1984, 911-913.

[11] D. W. Coit, A. E. Smith, "Reliability optimization of series-parallel systems using a genetic algorithm," *IEEE Transactions on Reliability*, vol. 45, 1996, 254-260.

[12] D. W. Coit, A. E. Smith, D. M. Tate, "Adaptive penalty methods for genetic optimization of constrained combinatorial problems," *INFORMS Journal on Computing*, vol. 8, 1996, 173-182.

[13] L. Davis, D. Orvosh, A. Cox, Y. Qui, "A genetic algorithm for survivable network design," *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, 408-415.

[14] D. L. Deeter, A. E. Smith, "Heuristic optimization of network design considering all-terminal reliability," *Proceedings of the Reliability and Maintainability Symposium*, 1997, 194-199.

[15] D. L. Deeter, A. E. Smith, "Economic design of reliable networks," *IIE Transactions*, in print.

[16] B. Dengiz, F. Altiparmak, A. E. Smith, "Efficient optimization of all-terminal reliable networks using an evolutionary approach," *IEEE Transactions on Reliability*, vol. 46, 1997, 18-26.

[17] B. Dengiz, F. Altiparmak, A. E. Smith, "Local search genetic algorithm for optimization of highly reliable communications networks," in Thomas Baeck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms* (ICGA97), 1997, 650-657.

[18] B. Dengiz, F. Altiparmak, A. E. Smith, "Local search genetic algorithm for optimization of highly reliable communications networks," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 3, 1997, 179-188.

[19] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco, CA, 1979.

[20] F. Glover, M. Lee, J. Ryan, "Least-cost network topology design for a new service: an application of a tabu search," *Annals of Operations Research*, vol. 33, 1991, 351-362.

[21] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing, Reading, MA, 1989.

[22] J. E. Hopcroft, J. D. Ullman, "Set merging algorithms," *SIAM Journal of Computers*, vol. 2, 1973, 296-303.

[23] K. Ida, M. Gen, T. Yokota, "System reliability optimization of series-parallel systems using a genetic algorithm," *Proceedings of the 16th International Conference on Computers and Industrial Engineering*, 1994, 349-352.

[24] R. H. Jan, "Design of reliable networks," *Computers and Operations Research*, vol. 20, 1993, 25-34.

[25] R. H. Jan, F. J. Hwang, S. T. Cheng, "Topological optimization of a communication network subject to a reliability constraint," *IEEE Transactions on Reliability*, vol. 42, 1993, 63-70.

[26] G. Katona, "A theorem on finite sets," in *Theory of Graphs*, Academia Kiado, Budapest, 1968, 187-207.

[27] S. J. Koh, C. Y. Lee, "A tabu search for the survivable fiber optic communication network design," *Computers and Industrial Engineering*, vol. 28, 1995, 689-700.

[28] A. Konak, A. E. Smith, "A general upper bound for all-terminal network reliability and its uses," *Proceedings of the Industrial Engineering Research Conference*, Banff, Canada, May 1998, CD Rom.

[29] A. Konak, A. E. Smith, "An improved general upper bound for all-terminal network reliability," under revision for *IIE Transactions*.

[30] J. B. Kruskal, "The number of simplices in a complex," in *Mathematical Optimization Techniques*, University of California Press, Berkeley, CA, 1963, 251-278.

[31] T. Kumamoto, K. Tanaka, K. Inoue, "Efficient evaluation of system reliability by Monte Carlo method," *IEEE Transactions on Reliability*, vol. R-26, no. 5, 1977, 311-315.

[32] A. Kumar, R. M. Pathak, Y. P. Gupta, H. R. Parsaei, "A genetic algorithm for distributed system topology design," *Computers and Industrial Engineering*, vol. 28, 1995, 659-670.

[33] A. Kumar, R. M. Pathak, Y. P. Gupta, "Genetic algorithm based reliability optimization for computer network expansion", *IEEE Transactions on Reliability*, vol. 44, 1995, 63-72.

[34] D. L. Nel, C. J. Colbourn, "Combining Monte Carlo estimates and bounds for network reliability," *Networks*, vol. 20, 1990, 277-298.

[35] L. Painton , J. Campbell, "Genetic algorithms in optimization of system reliability," *IEEE Transactions on Reliability*, vol. 44, 1995, 172-178.

[36] S. Pierre, M.-A. Hyppolite, J.-M. Bourjolly, O. Dioume, "Topological design of computer communication networks using simulated annealing," *Engineering Applications of Artificial Intelligence*, vol. 8, 1995, 61-69.

[37] J. S. Provan, M. O. Ball, "The complexity of counting cuts and of computing the

probability that a graph is connected," *Siam Journal of Computing*, vol. 12, no 4, 1983, 777-788.

[38] L. G. Roberts, B. D. Wessler, "Computer network development to achieve resource sharing," in *Proceedings of the Spring Joint Computing Conference*, AFIPS, vol. 36, 1970, 543-599.

[39] D. A. Savic, G. A. Walters, "An evolution program for pressure regulation in water distribution networks," *Engineering Optimization*, vol. 24, 1995, 197-219.

[40] A. E. Smith, D. M. Tate, "Genetic optimization using a penalty function," *Proceedings of the 5th International Conference on Genetic Algorithms*, 1993, 499-505.

[41] C. Srivaree-ratana, A. E. Smith, 'Estimating all-terminal network reliability using a neural network," *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, October 1998, 4734-4740.

[42] C. Srivaree-ratana, A. E. Smith, "An effective neural network estimate of reliability for use during optimal network design," under review for *INFORMS Journal on Computing*.

[43] D. M. Tate, A. E. Smith, "A genetic approach to the quadratic assignment problem," *Computers and Operations Research*, vol. 22, no. 1, 1995, 73-83.

[44] A. N. Ventetsanopoulos, I. Singh, "Topological optimization of communication networks subject to reliability constraints," *Problem of Control and Information Theory*, vol. 15, 1986, 63-78.

[45] G. A. Walters, D. K. Smith, "Evolutionary design algorithm for optimal layout of tree networks," *Engineering Optimization*, vol. 24, 1995, 261-281.

[46] M. S. Yeh, J. S. Lin, W. C. Yeh, "New Monte Carlo method for estimating network reliability," *Proceedings of 16th International Conference on Computers & Industrial Engineering*, 1994, 723-726.

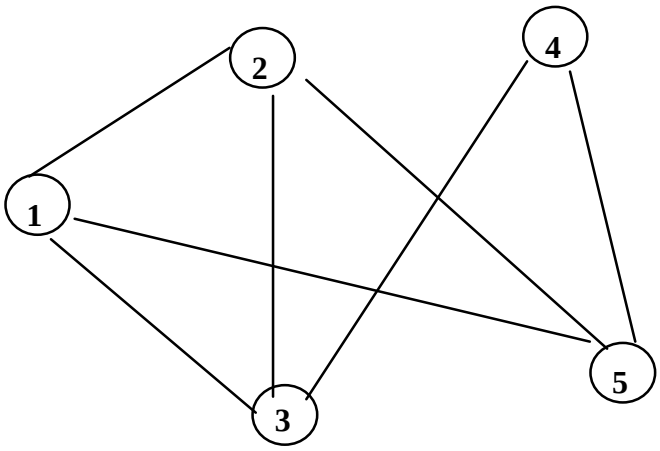
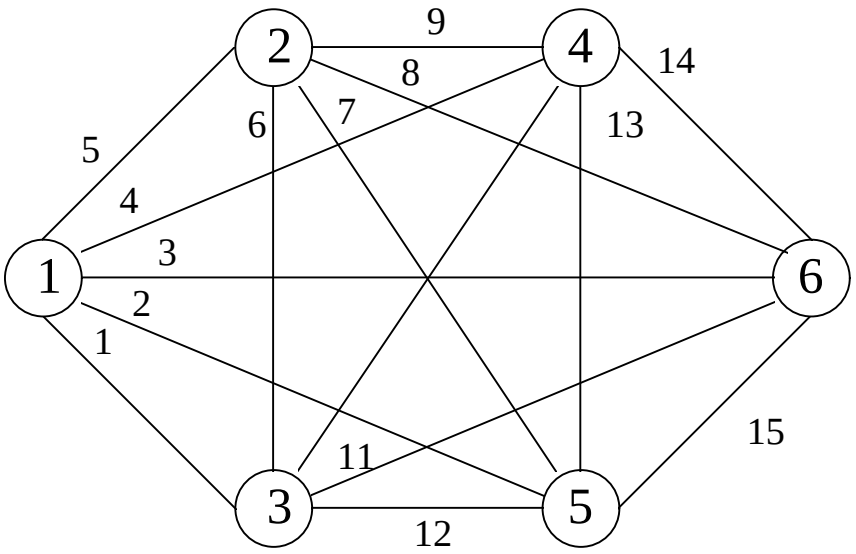
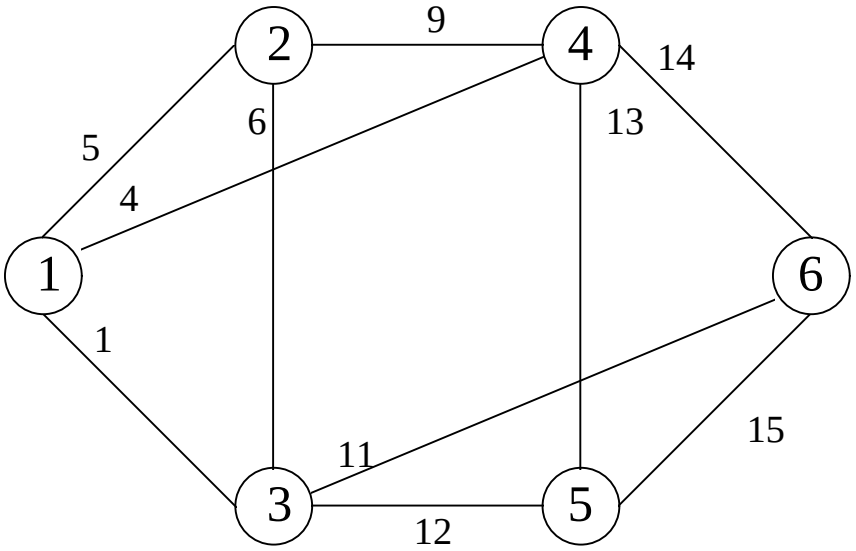


Figure 1. Five node network with arbitrarily numbered nodes.



2a. A fully connected network with 15 arcs that are arbitrarily labeled with integers 1 to 15.



2b. A partially connected network with 10 arcs using the same labeling scheme as in 2a.

Figure 2: Two networks with six nodes where arcs are arbitrarily labeled with integers 1 to 15.

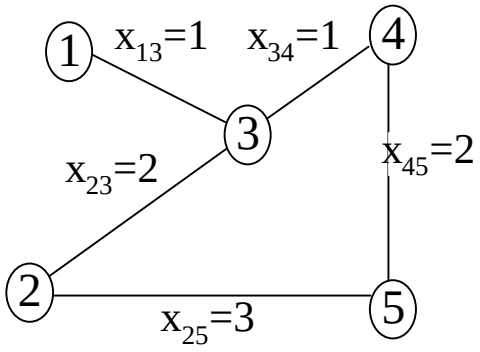


Figure 3. Example network design for section 6.

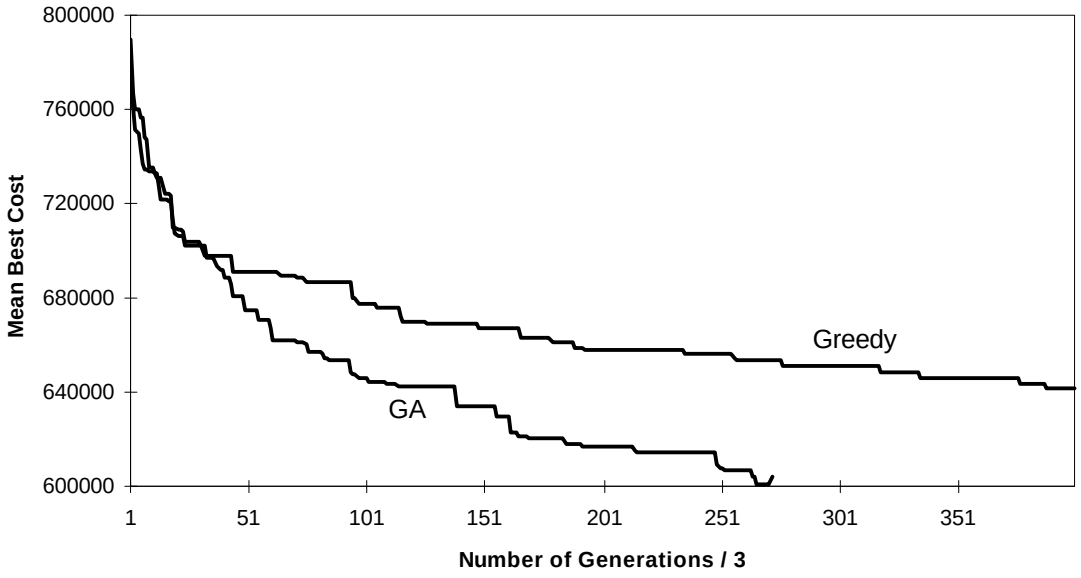


Figure 4. GA vs. greedy search averaged over 10 runs for the problem of section 6.2.

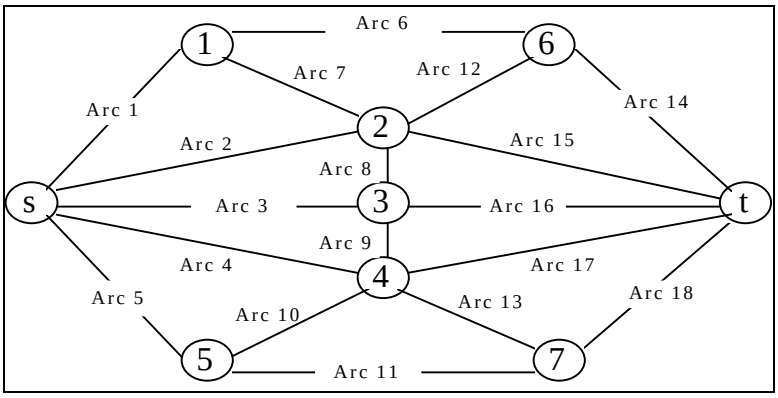


Figure 5. Source-sink problem topology of section 6.3.

.

Table 1: Comparison of results of GA from section 4.

Problem	N	L	p	R_o	Optimum Cost*	Results of Genetic Algorithm[@]		
						Best Cost	Mean Cost	Coef. of Variation
FULLY CONNECTED NETWORKS								
1	5	10	0.80	0.90	255	255	255.0	0
2	5	10	0.90	0.95	201	201	201.0	0
3	7	21	0.90	0.90	720	720	720.0	0
4	7	21	0.90	0.95	845	845	857.0	0.0185
5	7	21	0.95	0.95	630	630	656.0	0.0344
6	8	28	0.90	0.90	203	203	205.4	0.0198
7	8	28	0.90	0.95	247	247	249.5	0.0183
8	8	28	0.95	0.95	179	179	180.3	0.0228
9	9	36	0.90	0.90	239	239	245.1	0.0497
10	9	36	0.90	0.95	286	286	298.2	0.0340
11	9	36	0.95	0.95	209	209	227.2	0.0839
12	10	45	0.90	0.90	154	156	169.8	0.0618
13	10	45	0.90	0.95	197	205	206.6	0.0095
14	10	45	0.95	0.95	136	136	150.4	0.0802
15	15	105	0.90	0.95	---	317	344.6	0.0703
16	20	190	0.95	0.95	---	926	956.0	0.0304
17	25	300	0.95	0.90	---	1606	1651.3	0.0243
NON FULLY CONNECTED NETWORKS								
18	14	21	0.90	0.90	1063	1063	1076.1	0.0129
19	16	24	0.90	0.95	1022	1022	1032.0	0.0204
20	20	30	0.95	0.90	596	596	598.6	0.0052

* Found by the method of Jan et al. [25].

[@] Over ten runs.

Table 2: Comparison of search effort and reliability estimation of GA of section 4.

Problem	Search Space	Solutions Searched	Fraction Searched	R_o	Actual R(**x**)	Estimated R(**x**)	Percent Difference
1	1.02 E3	6.00 E2	5.86 E-1	0.90	0.9170	0.9170	0.000
2	1.02 E3	6.00 E2	5.86 E-1	0.95	0.9579	0.9604	0.261
3	2.10 E6	1.50 E4	7.14 E-3	0.90	0.9034	0.9031	-0.033
4	2.10 E6	1.50 E4	7.14 E-3	0.95	0.9513	0.9580	0.704
5	2.10 E6	1.50 E4	7.14 E-3	0.95	0.9556	0.9569	0.136
6	2.68 E8	2.00 E4	7.46 E-5	0.90	0.9078	0.9078	0.000
7	2.68 E8	2.00 E4	7.46 E-5	0.95	0.9614	0.9628	0.001
8	2.68 E8	2.00 E4	7.46 E-5	0.95	0.9637	0.9645	0.083
9	6.87 E10	4.00 E4	5.82 E-7	0.90	0.9066	0.9069	0.033
10	6.87 E10	4.00 E4	5.82 E-7	0.95	0.9567	0.9545	-0.230
11	6.87 E10	4.00 E4	5.82 E-7	0.95	0.9669	0.9668	-0.010
12	3.52 E13	8.00 E4	2.27 E-9	0.90	0.9050	*	
13	3.52 E13	8.00 E4	2.27 E-9	0.95	0.9516	*	
14	3.52 E13	8.00 E4	2.27 E-9	0.95	0.9611	0.9591	-0.208
15	4.06 E31	1.40 E5	3.45 E-27	0.95	@	0.9509	
16	1.57 E57	2.00 E5	1.27 E-52	0.95	@	0.9925	
17	2.04 E90	4.00 E5	1.96 E-85	0.90	@	0.9618	
18	2.10 E6	1.50 E4	7.14 E-3	0.90	0.9035	0.9035	0.000
19	1.68 E7	2.00 E4	1.19 E-3	0.95	0.9538	0.9550	0.126
20	1.07 E9	3.00 E4	2.80 E-5	0.90	0.9032	0.9027	-0.055

* Optimal not found by GA.

@ Network is too large to exactly calculate reliability.

Table 3: Complete results comparing performance and CPU time on 79 test problems.

Problem						B+B [25]	Section 4 GA		Section 5 GA	
No	*N*	*L*	*p*	R_O	Best Cost	CPU sec.	Coeff. Var. *	CPU sec.	Coeff. Var.*	CPU sec.
FULLY CONNECTED NETWORKS										
1	6	15	0.90	0.90	231	1.87	0.0245	57.50	0	11.97
2	6	15	0.90	0.90	239	0.01	0	41.05	0	8.28
3	6	15	0.90	0.90	227	0.04	0	38.90	0	12.30
4	6	15	0.90	0.90	212	0.17	0	46.32	0	12.60
5	6	15	0.90	0.90	184	0.28	0	52.39	0.0233	13.72
6	6	15	0.90	0.95	254	0.11	0	69.39	0.0217	19.48
7	6	15	0.90	0.95	286	0.00	0	50.17	0	13.04
8	6	15	0.90	0.95	275	0.06	0	48.37	0	12.40
9	6	15	0.90	0.95	255	0.06	0	59.32	0	14.36
10	6	15	0.90	0.95	198	0.01	0	53.65	0.0121	21.51
11	6	15	0.95	0.95	227	3.90	0.0357	57.98	0.0023	14.08
12	6	15	0.95	0.95	213	0.11	0.0235	47.83	0.0193	10.03
13	6	15	0.95	0.95	190	0.00	0.0280	42.32	0	10.09
14	6	15	0.95	0.95	200	0.44	0.0238	57.54	0.0173	13.04
15	6	15	0.95	0.95	179	0.66	0.0193	46.97	0.0256	11.36
16	7	21	0.90	0.90	189	11.26	0.0177	130.71	0.0175	21.77
17	7	21	0.90	0.90	184	0.17	0	76.74	0	18.80
18	7	21	0.90	0.90	243	0.50	0.0167	135.98	0.0202	26.93
19	7	21	0.90	0.90	129	1.21	0.0121	122.46	0.0195	28.91
20	7	21	0.90	0.90	124	0.05	0	83.45	0	23.77
21	7	21	0.90	0.95	205	0.83	0.0406	301.41	0.0337	71.40
22	7	21	0.90	0.95	209	0.06	0	4	0	37.06
23	7	21	0.90	0.95	268	0.06	0.0310	255.73	0.0187	56.39
24	7	21	0.90	0.95	143	0.17	0.0264	280.26	0.0193	78.72
25	7	21	0.90	0.95	153	0.01	0	160.43	0	52.93
26	7	21	0.95	0.95	185	22.85	0.0333	112.26	0.0111	28.89
27	7	21	0.95	0.95	182	1.27	0.0046	81.78	0.0035	16.99
28	7	21	0.95	0.95	230	1.76	0.0090	109.47	0.0072	26.64
29	7	21	0.95	0.95	122	2.31	0.0265	112.62	0.0259	27.82
30	7	21	0.95	0.95	124	0.39	0	74.49	0	19.64
31	8	28	0.90	0.90	208	21.9	0.0211	260.86	0.0161	79.55
32	8	28	0.90	0.90	203	20.37	0	175.06	0	75.37
33	8	28	0.90	0.90	211	140.66	0.0149	198.80	0.0119	79.67
34	8	28	0.90	0.90	291	173.01	0.0204	210.95	0.0108	83.66
35	8	28	0.90	0.90	178	159.34	0.0112	230.70	0	67.34
36	8	28	0.90	0.95	247	10162.53	0.0152	611.28	0.0140	168.79
37	8	28	0.90	0.95	247	15207.83	0.0274	808.94	0.0183	226.08
38	8	28	0.90	0.95	245	12712.21	0.0124	663.99	0.0034	184.31
39	8	28	0.90	0.95	336	9616.80	0.0169	743.39	0.0177	303.50
40	8	28	0.90	0.95	202	9242.10	0.0231	629.13	0.0235	266.47

* Over 10 runs.

Table 3 continued: Complete results comparing performance and CPU time on 79 test problems.

Problem						B+B [25]	Section 4 GA		Section 5 GA	
No	*N*	*L*	*p*	R_0	Best Cost	CPU sec.	Coeff. Var.*	CPU sec.	Coeff. Var.*	CPU sec.
FULLY CONNECTED NETWORKS										
41	8	28	0.95	0.95	179	0.11	0	133.32	0	43.81
42	8	28	0.95	0.95	194	2.69	0.0053	202.57	0.0033	40.56
43	8	28	0.95	0.95	197	26.97	0.0052	173.74	0.0080	58.04
44	8	28	0.95	0.95	276	20.76	0.0133	187.02	0.0100	50.64
45	8	28	0.95	0.95	173	72.78	0.0190	189.02	0.0206	53.51
46	9	36	0.90	0.90	239	8.02	0.0105	324.38	0.0066	98.19
47	9	36	0.90	0.90	191	23.78	0.0277	365.31	0.0081	153.77
48	9	36	0.90	0.90	257	702.05	0.0301	530.37	0.0171	176.79
49	9	36	0.90	0.90	171	0.82	0.0255	292.01	0	81.18
50	9	36	0.90	0.90	198	12.36	0.0228	378.91	0	90.49
51	9	36	0.90	0.95	286	8321.87	0.0821	1215.28	0.0325	404.93
52	9	36	0.90	0.95	220	14259.48	0.0330	998.79	0.0309	358.28
53	9	36	0.90	0.95	306	9900.87	0.0313	1256.82	0.0163	560.89
54	9	36	0.90	0.95	219	17000.04	0.0457	865.38	0.0226	340.13
55	9	36	0.90	0.95	237	7739.99	0.0760	1024.77	0.0778	391.52
56	9	36	0.95	0.95	209	4.95	0.0576	274.83	0	59.24
57	9	36	0.95	0.95	171	21.75	0.0137	293.43	0.0092	99.98
58	9	36	0.95	0.95	233	525.03	0.0375	372.18	0.0268	97.95
59	9	36	0.95	0.95	151	0.99	0.0471	252.71	0	65.78
60	9	36	0.95	0.95	185	25.92	0.0381	385.59	0	71.67
61	10	45	0.90	0.90	131	4623.19	0.0518	1047.60	0.0231	375.14
62	10	45	0.90	0.90	154	2118.75	0.0651	794.83	0.0223	214.63
63	10	45	0.90	0.90	267	1860.74	0.0142	999.01	0.0061	415.53
64	10	45	0.90	0.90	263	1466.73	0.0126	678.02	0	171.04
65	10	45	0.90	0.90	293	2212.70	0.0329	1093.36	0.0182	488.12
66	10	45	0.90	0.95	153	5712.97	0.0257	1718.45	0.0150	982.98
67	10	45	0.90	0.95	197	7728.21	0.0203	1689.51	0.0177	726.31
68	10	45	0.90	0.95	311	8248.16	0.0367	1967.61	0.0136	984.30
69	10	45	0.90	0.95	291	6802.16	0.0404	1529.61	0.0244	825.45
70	10	45	0.90	0.95	358	12221.39	0.0276	2662.34	0.0048	1071.99
71	10	45	0.95	0.95	121	3492.17	0.0563	793.22	0.0124	177.31
72	10	45	0.95	0.95	136	1125.89	0.0291	615.29	0.0185	81.87
73	10	45	0.95	0.95	236	987.64	0.0276	781.68	0.0160	139.53
74	10	45	0.95	0.95	245	2507.89	0.0369	632.11	0	98.31
75	10	45	0.95	0.95	268	1359.91	0.0513	630.37	0.0120	131.55
76	11	55	0.90	0.90	246	59575.49	0.0499	1532.34	0	472.11
NON FULLY CONNECTED NETWORKS										
77	14	21	0.90	0.90	1063	23950.01	0.0129	7293.97	0.0079	1672.75
78	16	24	0.90	0.95	1022	131756.43	0.0204	2699.38	0.0185	2334.15
79	20	30	0.95	0.95	596	#	0.0052	5983.24	0.0152	4458.81

* Over 10 runs.

\# Optimum solution taken from [25]. CPU time unknown.

Table 4. Arc unit costs and reliabilities for problems in section 6.

Connection Type (*k*)	Reliability	Unit Cost
not connected, 0	0.00	0
1	0.70	8
2	0.80	10
3	0.90	14