Local Search Genetic Algorithm for Optimal Design of Reliable Networks

Berna Dengiz and Fulya Altiparmak
Department of Industrial Engineering
Gazi University
06570 Maltepe, Ankara Turkey
berna@rorqual.cc.metu.edu.tr

Alice E. Smith, *Senior Member*, IEEE[1]
Department of Industrial Engineering
University of Pittsburgh
Pittsburgh, Pennsylvania 15261 USA
aesmith@engrng.pitt.edu

Abstract — This paper presents a genetic algorithm (GA) with specialized encoding, initialization and local search operators to optimize the design of communication network topologies. This NP-hard problem is often highly constrained so that random initialization and standard genetic operators usually generate infeasible networks. Another complication is that the fitness function involves calculating the all-terminal reliability of the network, a calculation that is computationally expensive. Therefore, it is imperative that the search balances the need to thoroughly explore the boundary between feasible and infeasible networks, along with calculating fitness on only the most promising candidate networks. The algorithm results are compared to optimum results found by branch and bound and also to GA results without local search operators on a suite of 79 test problems. This strategy of employing bounds, simple heuristic checks, and problem-specific repair and local search operators can be used on other highly constrained combinatorial applications where numerous fitness calculations are prohibitive.

Accepted to *IEEE Transactions on Evolutionary Computation*

August 1997

[1] Corresponding author.

Local Search Genetic Algorithm for Optimal Design of Reliable Networks

Abstract — This paper presents a genetic algorithm (GA) with specialized encoding, initialization and local search operators to optimize the design of communication network topologies. This NP-hard problem is often highly constrained so that random initialization and standard genetic operators usually generate infeasible networks. Another complication is that the fitness function involves calculating the all-terminal reliability of the network, a calculation that is computationally expensive. Therefore, it is imperative that the search balances the need to thoroughly explore the boundary between feasible and infeasible networks, along with calculating fitness on only the most promising candidate networks. The algorithm results are compared to optimum results found by branch and bound and also to GA results without local search operators on a suite of 79 test problems. This strategy of employing bounds, simple heuristic checks, and problem-specific repair and local search operators can be used on other highly constrained combinatorial applications where numerous fitness calculations are prohibitive.

Index Terms — genetic algorithm, local search, network reliability, network design, repair, penalty function, Monte Carlo simulation.

1 INTRODUCTION

Although the topological optimization of networks is an important problem in many fields such as telecommunications, electricity distribution and gas pipelines, it has major importance in the computer communication industry, when considering network reliability. In a communication network, *all-terminal* network reliability (also called *uniform* or *overall* network reliability) is defined as the probability that every pair of nodes can communicate with each other [1, 2]. This means that the network forms at least a spanning tree. The primary design problem is to choose a set of links for a given set of nodes, to either maximize reliability given a cost constraint, or to minimize cost given a minimum network reliability constraint. This design problem is NP-hard [3], and as a further complication, the calculation of all-terminal reliability is also NP-hard.

This problem and related versions have been studied in the literature with both enumerative-based methods and heuristic methods. Jan et al. [4] developed an algorithm using decomposition

based on branch and bound to minimize link 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. [5] maximized reliability given a cost constraint for networks with differing link reliabilities and an all-terminal reliability metric. Ventetsanopoulos and Singh [6] 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 [7] with exact calculation of network reliability to find the optimal design of very small networks (5 nodes or less). Pierre et al. [8] 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. [9] to choose network design when considering cost and capacity, but not reliability. Another tabu search approach by Beltran and Skorin-Kapov [10] was used to design reliable networks by searching for the least cost spanning 2-tree, where the 2-tree objective was a coarse surrogate for reliability. Koh and Lee [11] also used tabu search to find telecommunication network designs that required some nodes (special offices) have more than one link while others (regular offices) required only one link, also using this link 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 [12-14]. For network design, Kumar et al. [15] 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 links between any

two nodes) as a constraint. The same authors used this GA to expand existing computer networks [16]. Davis et al. [17] approached a related problem considering link capacities and re-routing upon link failure using a customized GA. Abuali et al. [18] assigned terminal nodes to concentrator sites to minimize costs while considering capacities using a GA, but no reliability was considered. The same authors in [19] solved the probabilistic minimum spanning tree problem where inclusion of the node in the network is stochastic and the objective is to minimize connection (link) costs, again without regard to reliability. Walters and Smith [20] 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, making this a somewhat simplistic approach. Deeter and Smith [21] presented a GA approach for a small (5 nodes) minimum cost network design problem with alternative link reliabilities and an all-terminal network reliability constraint. Dengiz et al. [22] all addressed the all-terminal network design problem on a test suite of 20 problems using a fairly standard GA implementation, and that method will be considered later in this paper. A shorter, earlier version of the research presented in this paper appeared in [23].

Given the NP-hard nature of the problem, heuristics are often needed to solve problems of realistic size. However, GAs have not been used as much as might be expected because of the difficulty of dealing with the feasibility issue. Highly reliable networks imply a severely constrained problem when minimum system reliability is used as a constraint. It is unknown whether or not a network is feasible until the network reliability is calculated. This calculation, if done exactly, is also NP-hard [24]. An alternative approach is to maximize network reliability given a maximum cost constraint, and in this case, network reliability must be calculated as part of the objective function. Table 1 shows the growth of the search space for both the design problem

(choice of links) and the exact calculation of network reliability (spanning trees and minimum cutsets). For networks of larger size, all-terminal reliability can be accurately estimated using a Monte Carlo simulation approach. While computationally tractable for large networks, Monte Carlo is nevertheless an expensive procedure for accurate estimation, from the standpoint of computational effort.

Insert Table 1 here.

The contributions of this paper are twofold. First, a difficult and realistic problem class is solved effectively and efficiently using a test suite of 79 problems. Previous work, including those cited above, have demonstrated the heuristic and exact optimization procedures on a small number of problems of limited network size, thus the important issue of scale-up is left unanswered. The 79 randomly generated test problems in this paper range up to 20 nodes and 55 possible links. Second, a general approach to employing easily calculated fitness surrogates to minimize the actual fitness calculation is married with local search and repair algorithms, a penalty function, and a seeding strategy to encourage the production of highly fit, feasible solutions. This is a good example of customizing the GA meta-heuristic to a highly constrained combinatorial problem where the fitness calculation is difficult. Local search proves more efficient in identifying near optimal solutions, thereby minimizing the fitness calculation.

2 STATEMENT OF THE PROBLEM

A communication network can be modeled by a probabilistic graph G = (N, L, p), in which N and L are the set of nodes and links that corresponds to the computer sites and communication connections respectively, and p is the connection (link) reliability. The networks are assumed to have bi-directional links and therefore are modeled by graphs with non-directed links. It is further assumed that the graph has no parallel (i.e. redundant) edges. Redundant links can be added to

improve reliability, and the approach described in this paper could be modified straightforwardly to include redundancy. The optimization problem is:

$$\text{Minimize} \qquad Z = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij}\, x_{ij} \tag{1}$$

$$\text{Subject to :} \qquad R(\mathbf{x}) \geq R_o$$

where $x_{ij} \in \{0,1\}$ is the decision variable, c_{ij} is the cost of (i,j) link, $R(\mathbf{x})$ is the network reliability and R_o is the minimum reliability requirement.

The following define the other problem assumptions:

1. The location of each network node is given.
2. Nodes are perfectly reliable.
3. Each c_{ij} and p are fixed and known.
4. Links are either operational or failed.
5. The failures of links are independent.
6. No repair is considered.

3 THE GENETIC ALGORITHM

3.1 ENCODING

A variable-length integer string representation was used following [25] to represent a water distribution system. Thiel et al. [26] also used this encoding to represent the possible insertion sequences of objects in a knapsack problem. Every possible link is assigned an integer and the presence of that link is signaled by the presence of that integer in the ordered string. The scheme for the integer assignment is arbitrary. The fully connected network in Figure 1 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 1 here.

String representations of networks given in Figure 1 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 links using the arbitrarily assigned labels defined above. The second network contains ten links, using the same labeling scheme. Node degree is defined as the number of links which emanate from a given node. For example, node 2 of the right-hand network of Figure 1 has node degree = 3.

3.2 INITIAL POPULATION

To enhance the efficiency of the search, the initial population consists of networks with the characteristics of being highly reliable. The combination of a stochastic depth-first algorithm with repair is used to generate the initial population by:

1. A spanning tree is implemented through the depth-first search algorithm by Hopcroft and Ullman [27], which grows a tree from a randomly chosen node.
2. Links selected randomly from the co-tree set (the set of links 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 [28], it is repaired by the algorithm explained in section 3.5.

3.3 OBJECTIVE FUNCTION

The objective function is the sum of the total cost for all links in the network 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 optimization algorithm to near-optimal, feasible solutions. It is important to allow infeasible solutions into the population because good solutions can be the result of breeding between feasible and infeasible solutions, and the reproduction procedure does not ensure feasible children even if both parents are feasible, especially in highly constrained problems where the constraint is likely to be active. There has been a body of work published in evolutionary computation on handling constraints (the most recent comprehensive treatment is found in [29]). In particular, Michalewicz [30-33] and Smith [34-35] have worked on using penalty functions to effectively and efficiently guide evolutionary search to feasible, optimal (or near-optimal) final solutions. The penalty function below uses the notion of distance of the solution from feasibility (the $R(\mathbf{x})$-R_o term) and a nonlinear penalty (the exponent of 2).

The fitness function is given by,

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

$$\delta = \begin{cases} 0, & \text{if } R(\mathbf{x}) \geq R_o \\ 1, & \text{if } R(\mathbf{x}) < R_o \end{cases}$$

c_{MAX} = the maximum value of c_{ij}

For computation of $R(\mathbf{x})$, three reliability estimations are used to trade off accuracy with computational effort. An ideal strategy would only employ the computationally intensive method

of Monte Carlo simulation on the optimal network design. Since the GA is an iterative algorithm, this ideal cannot be attained as many candidate networks must be evaluated during the search. Therefore, screening of candidate network designs is used. First, a connectivity check for a spanning tree is made on all new network designs using the method of [27]. Then, for networks which pass this check, the 2-connectivity measure of [28] is made by ensuring that all nodes have at least degree 2. Finally, for networks which pass both of these preliminary checks, Jan's upper bound [2] is used to compute the upper bound of reliability of the candidate network, $R_U(\mathbf{x})$. This upper bound is used in the calculation of the objective function (eq. 2) for all networks except those which are the best found so far ($\mathbf{x}_{BEST}$). Networks which have $R_U(\mathbf{x}) \geq R_0$ and the lowest cost so far are sent to the simulation subroutine for precise estimation of network reliability using an efficient Monte Carlo technique by Yeh et al. [36]. This Monte Carlo technique improves upon the classic method by reducing the variability of the estimate of network reliability, allowing for a more efficient estimator.

3.4 THE ALGORITHM

The flow of the algorithm is as follows:

Step 1: Generate the initial population of size *s* by the method of section 3.2. Calculate the fitness of each candidate network in the population using eq. 2 and Jan's upper bound [2] as $R(\mathbf{x})$, 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 $R(\mathbf{x})$ in eq. 2. Generation, g, = 1.

Step 2: Select two candidate networks. An elitist ranking selection with stochastic remainder sampling without replacement is used [37].

Step 3: To obtain two children, apply crossover (defined in 3.6) to the selected networks and mutation (defined in 3.7) to the children.

Step 4: Determine the 2-connectivity of each new child. Use the repair algorithm (defined in 3.5) 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 $g = g+1$.

3.5 2-CONNECTIVITY REPAIR ALGORITHM

If any candidate network does not pass 2-connectivity (i.e., has one or more nodes with node degree < 2), the network is repaired using three different alternatives according to how many nodes fail the test. The repair strategy is basically a greedy link addition procedure.

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 connection between this node and the nodes in the N_{min} set has minimum cost and connect them, 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}

An illustrative example of the connectivity repair algorithm is presented. A candidate network with 5 nodes and link costs of

$c_{1,2}$=32, $c_{1,3}$=54, $c_{1,4}$=62, $c_{1,5}$=25, $c_{2,3}$=34, $c_{2,4}$=58, $c_{2,5}$=45, $c_{3,4}$=36,$c_{3,5}$=52, $c_{4,5}$=29

is shown in Figure 2a.

Insert Figure 2 here.

Step 1: N_1=[1,2] , n_1=2; N_2=[3,4,5], n_2=3. In this case; $N_{min} = N_2$ and $n_{min} = n_2$. Apply step 2b, because $n_1 = 2$.

Step 2b: 1 and 2 are the nodes in the N_1 set; 3, 4, 5 are the nodes in the N_{min} set. The connection cost of the two nodes in N_1 set is $c_{1,2}$=32. The connection costs of m_{11} to nodes in the N_{min} set are $c_{1,3}$=54, $c_{1,4}$=62 and $c_{1,5}$ already exists. $c_{1,3} < c_{1,4}$; $c_{1,3}$ is the minimum cost connection. The

connection costs of m_{12} to nodes in the N_{min} set are $c_{2,3}$=34, $c_{2,4}$ already exists and $c_{2,5}$=45. Since $c_{2,3} < c_{2,5}$ $c_{2,3}$ is the minimum connection. $c_{1,2} < [c_{1,3} + c_{2,3}]$, so node 1 is connected to node 2. After repair, the network shown in Figure 2b is obtained.

3.6 CROSSOVER OPERATOR

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 links = T1∩ T2, other links are: $\overline{T1}$ = T1 - (T1∩T2) ; $\overline{T2}$ = T2 - (T1∩T2)

Step 2: Assign common links 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: Links 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 links of T1 ∪ T2 do not exist in T1′ and T2′: CT1 = T1 \ T1′; CT2 = T2 \ T2′

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

An illustrative example of the crossover operator is shown. Figures 3a and 3b show the selected T1 and T2 parents. All link costs are the same as for the network in Figure 2. Note that the encoding would be the integer link labeling as in Figure 1, however for clarity in the example below, the links are labeled by the nodes they connect.

Insert Figure 3 here.

Step 1: T1∩T2=[1,3; 2,4; 3,5]; $\overline{T1}$=[1,2; 2,3; 4,5] , $\overline{T2}$ =[1,5; 2,5; 3,4].

Step 2: Assign common links to children as shown in Figures 4a and 4b.

Insert Figure 4 here.

Step 3: T1′ and T2′ are not spanning trees.

Step 4: c_{ij} in $\overline{\text{T1}}$: $c_{1,2}$=32, $c_{2,3}$=34, $c_{4,5}$=29 is minimum.; c_{ij} in $\overline{\text{T2}}$: $c_{2,5}$=45, $c_{3,4}$=36 is minimum. Add links 4,5 and 3,4 to make spanning trees.

Insert Figure 5 here.

Steps 5 and 6: Leftover links from each parent are added to the opposite children.

Insert Figure 6 here.

T2′ still has 1 degree for node 5, therefore the repair algorithm of section 3.5 is invoked and the final child network T2′ of Figure 7 results.

Insert Figure 7 here.

3.7 MUTATION OPERATOR

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 link not in the network and add it; stop.

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

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

go to step 3.

The mutation operator is illustrated assuming the network in Figure 8a and link costs as in Figure 2.

Insert Figure 8 here.

Step 1: In this network, the node degrees deg(j); j = 1,2,...,N are: deg(1)=2, deg(2)=2, deg(3)=3, deg(4)=2, deg(5)=3. As shown, the nodes 1, 2 and 4 have node degree=2 and 3 and 5 have node degree=3 and N_2 = [1, 2, 4], N_3 = [3, 5]; deg(j) ≥2. In this case, step 4 is applied.

Step 4: Generate $u \sim U(0,1)$; for example u = 0.4578 and DR=0.70. $u \geq$(1-DR), so go to step 3.

Step 3: Use drop operator. $c_{1,3}$ = 54, $c_{3,5}$ = 52, $c_{2,5}$ = 45, $c_{3,4}$ = 36, $c_{1,2}$ = 32, $c_{4,5}$ = 29. Dropping $L_{1,3}$ fails 2-connectivity, so drop $L_{3,5}$. The mutated network is shown in Figure 8b.

3.8 PARAMETER VALUES OF GA

Performance was systematically investigated for a set of five parameters: network size (NS), population size (PS), crossover rate (CR), mutation rate (MR), and drop rate (DR). Three levels were selected for each parameter, so the experimental design included 3^5 design points. Five replications were made for each design point, resulting in 1215 observations. Statistical analysis was performed using analysis of variance (ANOVA) and Duncan's multiple range tests and the results are shown in Table 2. While NS, PS, and MR were significant at α = 0.05, CR and DR were not. The F-statistic values for NS and PS were larger than that of MR, suggesting that the variations in the levels of NS and PS have greater impact on performance than does MR. It is not surprising that network size affects the search, or that the interaction between network size and population size is significant, because of the exponential increase in search space as each node is added. A few of the other two-way interactions were slightly significant. The best results were found for PS = 50 or 75, CR = 0.50, 0.60 or 0.70, MR = 0.20 or 0.30, and DR = 0.50, 0.60 or

0.70. In this paper, the parameters are set at PS = 50, CR = 0.70, MR = 0.30 and DR=0.60. The population size is somewhat small for conventional GAs, however it was chosen considering the computational effort needed to evaluate each solution. Since populations of 50 and 75 were not statistically significantly different, the lower value was chosen. Note that mutation is fairly active; this is a result of its local search effect which appears to fine tune promising search spaces identified by crossover.

Insert Table 2 here.

4 COMPUTATIONAL RESULTS

There are two comparisons made to judge the effectiveness and efficiency of the network GA with local search, termed LS/NGA. (Recall that repair, in the form of greedy local search, is done by both the crossover and mutation operators, and when generating the initial population.) These are the branch and bound (B+B) technique by Jan et al. [4] and the Network GA (NGA) that was fully investigated in [22]. NGA uses a binary encoding, single point crossover, and bit flip mutation; no repair or local search is performed. However, the fitness calculation (including the bounding and Monte Carlo simulation) is identical to LS/NGA, as are the selection mechanism, the penalty function, and the use of the 2-connectivity screen for initial population generation.

The 79 randomly generated test problems are both fully connected and non-fully connected networks with N ranging from 6 to 20[1]. The available links of the non-fully connected networks were randomly generated and were 1.5 times N. The link costs for all networks were randomly generated over [1, 100]. Each problem for the GAs was run ten times with different random number seeds to gauge the natural variability of the GA.

Table 3 shows a summary of the test problems comparing the performance of the two GA

[1] All test problems are available from the authors.

approaches with the optimal solutions, in terms of nearness to optimality and computational effort. The results are averaged over each problem instance of each network size and over the ten replications of each problem instance. It can be seen that LS/NGA does not degrade in performance with increase in problem size while NGA does. Furthermore, while computational effort grows with problem size, it is a more modest growth than for NGA and many orders of magnitude less than the exponential growth for enumerative based methods. This comparison of computational effort is more clearly seen in Table 4. All computational comparisons were made on a Pentium 133 MHz PC using Pascal code.

Insert Tables 3 and 4 here.

Table 5 lists complete results of the three methods for all 79 test problems. The conclusions of the results summarized in Tables 3 and 4 are confirmed. 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.

Insert Table 5 here.

Applying statistical tests to the results gives the following. Paired t-tests[2] between the coefficient of variation over 10 runs yields that LS/NGA is superior to NGA with a p-value of 0.0000 and a mean improvement (decrease in coefficient of variation) of 0.0104. The distributions of the CPU times of all three methods did not meet the requirements of efficient non-parametric or parametric statistical tests. A non-parametric sign test of CPU between LS/NGA and NGA resulted in a p-value < 0.0000 that LS/NGA is more efficient with a mean improvement of 392 seconds. A sign test of CPU of LS/NGA and B+B was inconclusive. For small problems, B+B is much more efficient, however it becomes orders of magnitude less efficient for large

[2] The residuals of the ten pairs were distributed approximately normally.

problems. This is typical computational behavior of a heuristic versus an enumerative method as search space grows exponentially.

5 CONCLUSIONS

It is not surprising that a special purpose GA is more efficient than an enumerative based method on NP-hard problems of realistic size. It is encouraging that the heuristic GA is very effective in identifying optimal solutions, even in search spaces up to 10^{16}. The problem studied, while being of interest in many real applications, is not one that particularly lends itself to an evolutionary approach at first glance. There are several major barriers which had to be overcome. First, the problem, when the network must be highly reliable, is very constrained. This is handled initially by repairing children to ensure they at least *might be* highly reliable. For networks which might be highly reliable, but are not (identified after network reliability is calculated) the infeasibility is handled via a distance-based quadratic exterior penalty function. Second, the fitness calculation is computationally burdensome, so use of bounds and repair and local search operators are used. Bounds serve as surrogates in the reliability fitness function for networks which are not the best candidates for the final solution. Repair and local search help identify networks which are particularly promising in their region of the search space.

What is of greater interest is the series of steps which can be incorporated into an evolutionary meta-heuristic, such as a GA, which enables the efficient and effective optimization of highly constrained problems with large search spaces where the calculation of fitness is difficult. The steps used included seeding the initial population with solutions that are prone to be highly fit, crossover and mutation operators which tend to produce highly fit offspring, and the judicious use of quickly calculated surrogates for fitness.

Repair operators and local search mechanisms will be problem specific. In this paper, they

are simple greedy operators that work by adding or subtracting the lowest or highest cost link. In other problems, similar uncomplicated approaches using the notion of neighboring solutions may work well. The primary objective is to use some problem-specific knowledge to craft simple mechanisms to encourage the production of solutions that are apt to be fit and feasible. To identify when the local search repair mechanisms are needed, fitness surrogates should be employed where possible. In this paper, first a connectivity check, then counting node degrees were applied to screen for highly reliable networks. For other problems, there may be somewhat crude, but reasonable, ways to quickly examine a solution for the likelihood of superior fitness. Finally, exact calculation of fitness is largely avoided by using an upper bound on all but the most superior candidates. Upper and lower bounds exist for many optimization problems, such as scheduling and routing. Depending on their tightness and their ease of calculation, these bounds may be valuable fitness surrogates during search and their usefulness should be exploited to craft an efficient evolutionary algorithm. It must be cautioned that use of surrogates and inexact fitness calculations may, in some cases, fail to allow the search to identify the optimal solution. However, since what is usually desired is a very good solution, rather than the single optimal solution, this possibility is more of an academic concern than a real one.

Acknowledgments

Part of this study was funded by The Government Planning Organization of Turkey (DPT), project number DPT-96K-120820. A. E. Smith is pleased to acknowledge the support of the U.S. National Science Foundation CAREER grant DMI 95-02134.

References

[1] C. J. Colbourn, *The Combinatorics of Network Reliability,* 1987; Oxford University Press.

[2] R. H. Jan, "Design of Reliable Networks", *Computers and Operations Research*, vol **20**,

1993, pp 25-34.

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

[4] 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, pp 63-70.

[5] K. K. Aggarwal, Y. C. Chopra, J. S. Bajwa, "Reliability Evaluation by Network Decomposition", *IEEE Transactions on Reliability*, vol **R-31**, 1982, pp 355-358.

[6] A. N. Ventetsanopoulos , I. Singh, "Topological Optimization of Communication Networks Subject to Reliability Constraints", *Problem of Control and Information Theory*, vol **15**, 1986, pp 63-78.

[7] M. M. Atiqullah, S. S. Rao, "Reliability Optimization of Communication Networks using Simulated Annealing", *Microelectronics and Reliability*, vol **33**, 1993, pp 1303-1319.

[8] 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, pp 61-69.

[9] 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, pp 351-362.

[10] H. F. Beltran, D. Skorin-Kapov, "On Minimum Cost Isolated Failure Imune Networks", *Telecommunications Systems*, vol **3**, 1994, pp 183-200.

[11] S. J. Koh, C. Y. Lee, "A Tabu Search for the Survivable Fiber Optic Communication Network Design", *Computers and Industrial Engineering*, vol **28**, 1995, pp 689-700.

[12] D. W. Coit, A. E. Smith, "Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm", *IEEE Transactions on Reliability*, vol **45**, 1996, pp 254-260.

[13] 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, pp 349-352.

[14] L. Painton , J. Campbell, "Genetic Algorithms in Optimization of System Reliability", *IEEE Transactions on Reliability*, vol **44**, 1995, pp 172-178.

[15] 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, pp 659-670.

[16] A. Kumar, R. M. Pathak, Y. P. Gupta, "Genetic Algorithm Based Reliability Optimization for Computer Network Expansion", *IEEE Transactions on Reliability*, vol **44**, 1995, pp 63-72.

[17] 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, pp 408-415.

[18] 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, pp 74-81.

[19] 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, pp 242-246.

[20] G. A. Walters, D. K. Smith, "Evolutionary Design Algorithm for Optimal Layout of Tree Networks", *Engineering Optimization*, vol **24**, 1995, pp 261-281.

[21] D. L. Deeter, A. E. Smith, "Heuristic Optimization of Network Design Considering All-Terminal Reliability", *Proceedings of the Reliability and Maintainability Symposium*, 1997, pp 194-199.

[22] 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, pp 18-26.

[23] 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, pp 650-657; East Lansing, MI.

[24] M. Ball, R. M. Van Slyke, "Backtracking Algorithms For Network Reliability Analysis", *Annals of Discrete Mathematics*, vol **1**, 1977, pp 49-64.

[25] D. A. Savic, G. A. Walters, "An Evolution Program for Pressure Regulation in Water Distribution Networks", *Engineering Optimization*, vol **24**, 1995, pp 197-219.

[26] J. Thiel, S. Voss, "Some Experiences on Solving Multiconstraint Zero-One Knapsack Problems with Genetic Algorithms", *INFOR Journal*, vol **32**, 1994, pp 226-242.

[27] J. E. Hopcroft, J. D. Ullman, "Set Merging Algorithms", *SIAM Journal of Computers*, vol **2**, 1973, pp 296-303.

[28] L. G. Roberts, B. D. Wessler, "Computer Network Development to Achieve Resource Sharing", in *Proceedings of the Spring Joint Computing Conference*, AFIPS Conf. Proc. 36, 1970, pp 543-599; AFIPS Press.

[29] T. Baeck, D. B. Fogel, Z. Michalewicz, *Handbook of Evolutionary Computation, Part C5*, Bristol, UK, 1997; Institute of Physics Publishing and Oxford University Press.

[30] Z. Michalewicz, "Genetic Algorithms Numerical Optimization and Constraints", *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995, pp 151-158.

[31] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs, 3rd edition*, New York, NY, 1996; Springer.

[32] Z. Michalewicz , G. Nazhiyath, "Genocop III: A Co-Evolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints", *Proceedings of the 2nd IEEE International Conference on Evolutionary Computation*, 1995, pp 647-651.

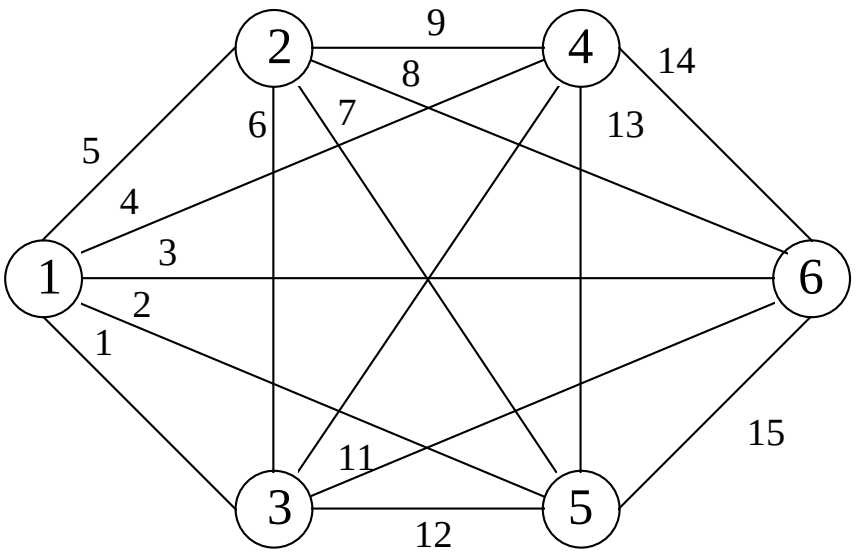
[33] Z. Michalewicz, N. Attia, "Evolutionary Optimization of Constrained Problems", *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, 1994, pp 98-108.

[34] A. E. Smith, D. M. Tate, "Genetic Optimization Using a Penalty Function", *Proceedings of the 5th International Conference on Genetic Algorithms*, 1993, pp 499-505.

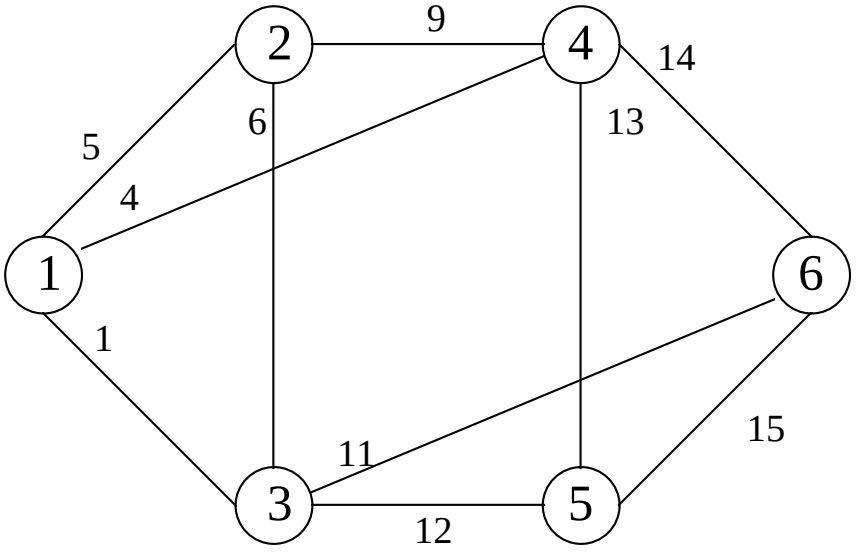
[35] D. W. Coit, A. E. Smith, D. M. Tate, "Adaptive Penalty Methods for Genetic Optimization of Constrained Combinatorial Problems", *INFORMS Journal on Computing*, vol **8**, 1996, pp 173-182.

[36] 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, pp 723-726.

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



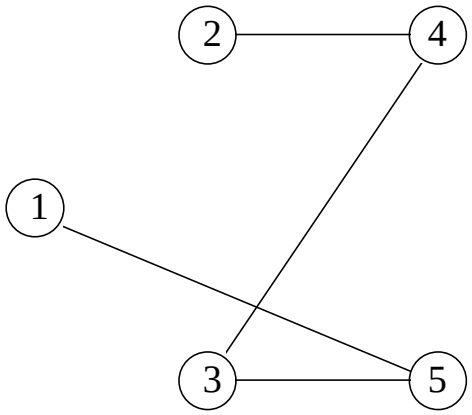
1a. A fully connected network with 15 links that are arbitrarily labeled with integers 1 to 15.



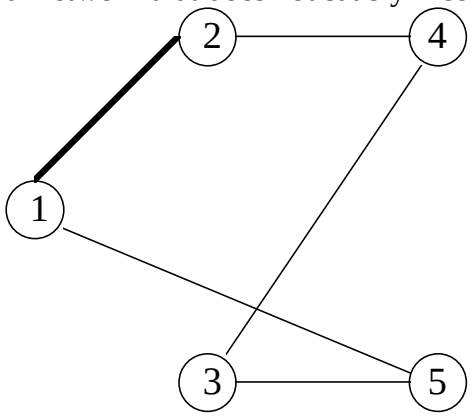
1b. A partially connected network with 10 links using the same labeling scheme as in 1a.

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

This labeling forms the encoding of the network for the GA.

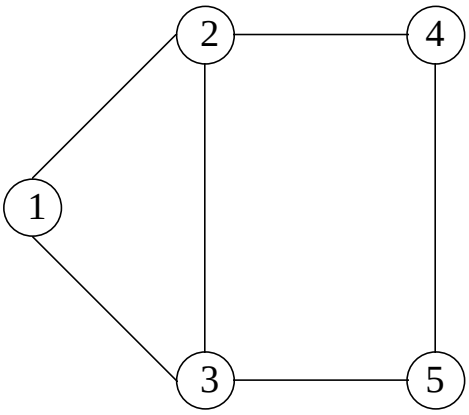


2a. Original network that does not satisfy 2-connectivity.

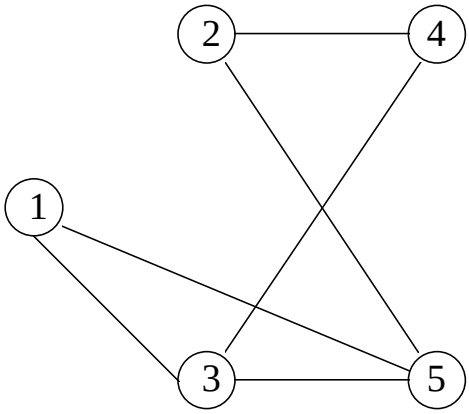


2b. Repaired network where the link between nodes 1 and 2 (in bold) is added.

Figure 2: A network with five nodes that is repaired for 2-connectivity by adding a link from node 1 to node 2.

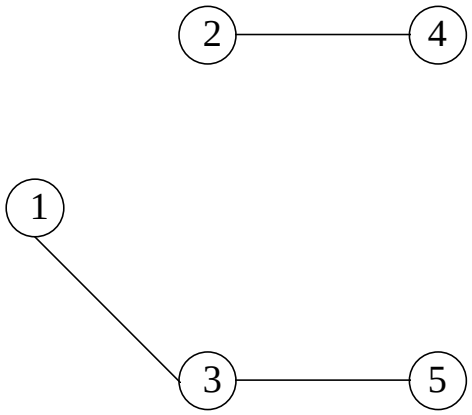


3a. Parent T1.

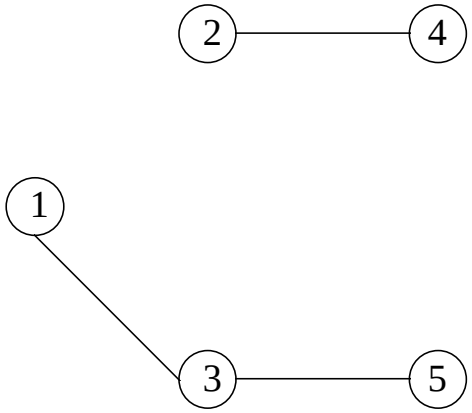


3b. Parent T2.

Figure 3: Two networks with five nodes that have been selected for crossover.

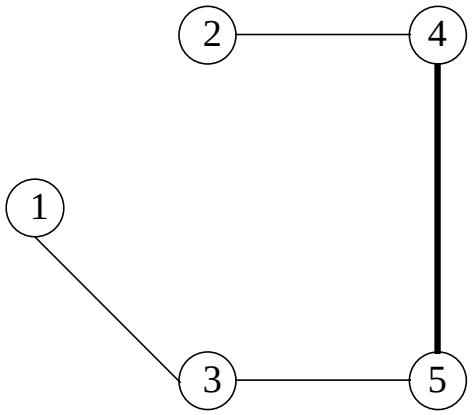


4a. Child T1´.

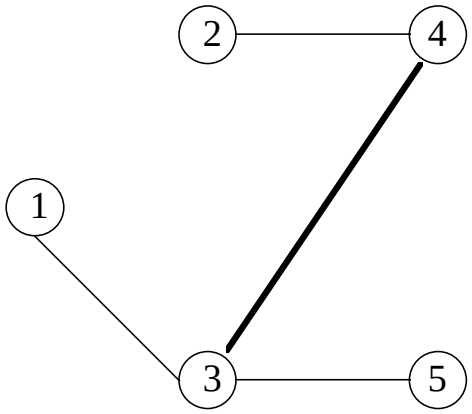


4b. Child T2´.

Figure 4: The initial step of creation of two children takes links common to both parents.

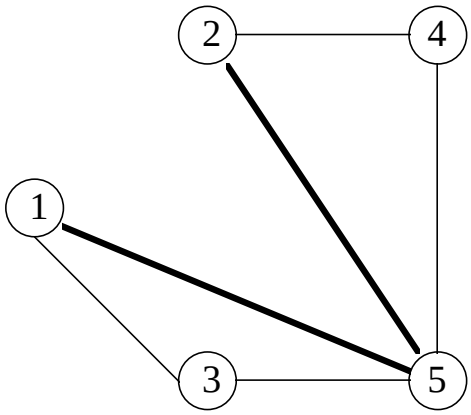


5a. A link between nodes 4 and 5 (in bold) is added to make a spanning tree T1´.

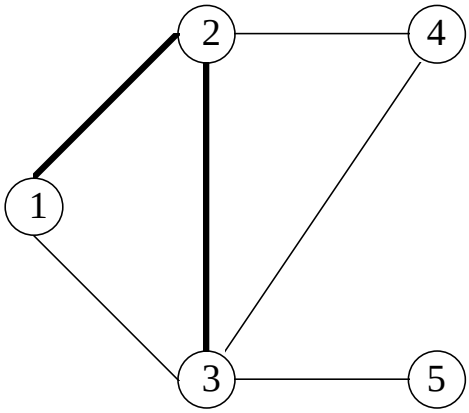


5b. A link between nodes 3 and 4 (in bold) is added to make a spanning tree T2´.

Figure 5: The second step of creation of two children that adds links from each parent (in bold) to make each child a spanning tree.



6a. Child T1′ is composed of T1′ (Figure 5a) ∪ CT2 (in bold).



6b. Child T2′ is composed of T2′ (Figure 5b) ∪ CT1 (in bold).

Figure 6: The final step in the creation of two children.

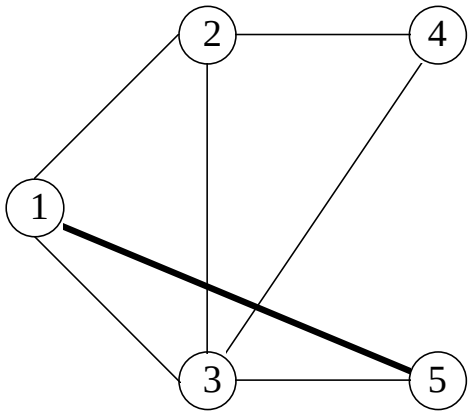
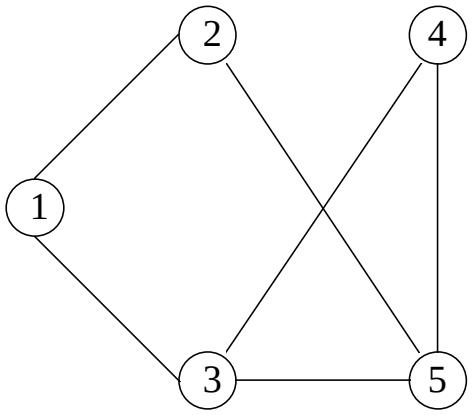
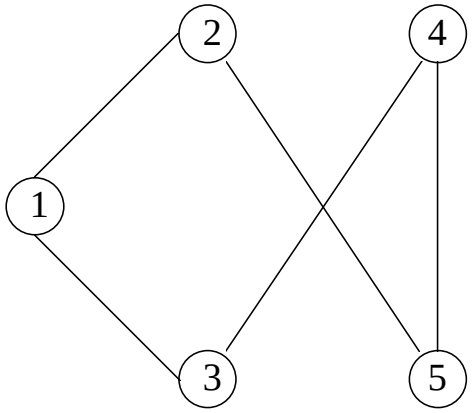


Figure 7: T2′ from Figure 6b that has undergone repair for 2-connectivity. The link between nodes 1 and 5 (in bold) has been added.



8a. Network with deg(j) ≥ 2 before mutation.



8b. Network after mutation where the link from node 3 to node 5 has been deleted.

Figure 8: Mutation of a network with five nodes and six links.

Table 1: Search space size for four network sizes.

# Nodes (*N*)	**7**	**10**	**15**	**20**
# Links ($L = \frac{N(N-1)}{2}$)	21	45	105	190
Search Space (2^L)	2.10×10^{6}	3.51×10^{13}	4.05×10^{31}	1.56×10^{57}
# Spanning Trees (N^{N-2})	1.68×10^{4}	1.00×10^{8}	1.94×10^{15}	2.62×10^{23}
# Minimum Cutsets ($2^N - 1$)	1.27×10^{2}	1.02×10^{3}	3.27×10^{4}	1.04×10^{6}

Table 2: ANOVA and Duncan's test results.

Source of variation	DF	F-values	P-values
Model	50	6.730	0.000
Error	192		
Corrected Total	242	R^2 = 0.667	
NS	2	59.475	0.000
PS	2	40.198	0.000
CR	2	1.746	0.177
MR	2	17.530	0.000
DR	2	2.713	0.069
NS x PS	4	9.845	0.000
NS x CR	4	0.758	0.554
NS x MR	4	2.761	0.029
NS x DR	4	0.926	0.450
PS x CR	4	0.922	0.453
PS x MR	4	4.003	0.004
PS x DR	4	0.882	0.476
CR x MR	4	0.345	0.847
CR x DR	4	0.736	0.586
MR x DR	4	1.315	0.266

Significant Factors	n	Group $\bar{x}$	Duncan grouping	Levels
NS	81	0.00192	B	7
	81	0.00266	B	8
	81	0.00846	A	10
PS	81	0.00761	A	25
	81	0.00319	B	50
	81	0.00216	B	75
MR	81	0.00636	A	0.10
	81	0.00406	B	0.20
	81	0.00260	B	0.30

Table 3: Summary of results of two GA approaches (averaged over 10 runs of each problem size).

Problem			NGA [22]		LS/NGA	
N	*L*	Search Space	Mean Solns Searched	Mean % from Optimal	Mean Solns Searched	Mean % from Optimal
6	15	3.28×10^{4}	2378	0.472	1596	0.400
7	21	2.09×10^{6}	6254	1.068	4190	0.777
8	28	2.68×10^{8}	11638	1.176	7811	0.889
9	36	6.87×10^{10}	28166	2.957	12922	1.050
10	45	3.15×10^{13}	62783	3.509	34168	1.094
11	55	3.60×10^{16}	83833	4.675	43566	0.323

Table 4: Comparison of computation time.

Problem		Mean CPU Seconds		
N	*L*	B+B [4]	NGA [22]	LS/NGA
6	15	0.514	51.313	13.216
7	21	2.859	145.741	35.775
8	28	3839.133	361.253	118.751
9	36	3903.195	588.717	203.386
10	45	4164.566	1175.533	458.937
11	55	59575.263	1532.341	472.105

Table 5: Complete results comparing performance and CPU time.

Problem						B+B [4]	NGA [22]		LS/NGA	
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 5 continued: Complete results comparing performance and CPU time.

Problem						B+B [4]	NGA [22]		LS/NGA	
No	*N*	*L*	*p*	R_O	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 [4]. CPU time unknown.

List of Figure Captions.

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

1b. A partially connected network with 10 links using the same labeling scheme as in 1a.

Figure 1: Two networks with six nodes where links are arbitrarily labeled with integers 1 to 15. This labeling forms the encoding of the network for the GA.

2a. Original network that does not satisfy 2-connectivity.

2b. Repaired network where the link between nodes 1 and 2 (in bold) is added.

Figure 2: A network with five nodes that is repaired for 2-connectivity by adding a link from node 1 to node 2.

3a. Parent T1.

3b. Parent T2.

Figure 3: Two networks with five nodes that have been selected for crossover.

4a. Child T1´.

4b. Child T2´.

Figure 4: The initial step of creation of two children takes links common to both parents.

5a. A link between nodes 4 and 5 (in bold) is added to make a spanning tree T1´.

5b. A link between nodes 3 and 4 (in bold) is added to make a spanning tree T2´.

Figure 5: The second step of creation of two children that adds links from each parent (in bold) to make each child a spanning tree.

6a. Child T1′ is composed of T1′ (Figure 5a) $\cup$ CT2 (in bold).

6b. Child T2′ is composed of T2′ (Figure 5b) $\cup$ CT1 (in bold).

Figure 6: The final step in the creation of two children.

Figure 7: T2′ from Figure 6b that has undergone repair for 2-connectivity. The link between nodes 1 and 5 (in bold) has been added.

8a. Network with $\deg(j) \geq 2$ before mutation.

List of Figure Captions, continued.

8b. Network after mutation where the link from node 3 to node 5 has been deleted.

Figure 8: Mutation of a network with five nodes and six links.

List of Table Captions.

Table 1: Search space size for four network sizes.

Table 2: ANOVA and Duncan's test results.

Table 3: Summary of results of two GA approaches (averaged over 10 runs of each problem size).

Table 4: Comparison of computation time.

Table 5: Complete results comparing performance and CPU time.

Table 5 continued: Complete results comparing performance and CPU time.

AUTHORS

Dr. Berna Dengiz; Department of Industrial Engineering; Gazi University; 06570 Maltepe, Ankara Turkey

Internet (e-mail): berna@rorqual.cc.metu.edu.tr

Berna Dengiz is an Associate Professor of Industrial Engineering and also Vice Dean of the Faculty of Engineering. Her field of study is the modeling and optimization of complex systems. Her research has been sponsored by TUBITAK - The Scientific and Technical Research Council of Turkey. Dr. Dengiz is a member of SCS - Society for Computer Simulations, Operations Research Society of Turkey, Informatics Society of Turkey and Statistics Society of Turkey.

Dr. Fulya Altiparmak; Department of Industrial Engineering; Gazi University; 06570 Maltepe, Ankara Turkey

Fulya Altiparmak is a Research Assistant of Industrial Engineering. She received a B.Sc. (1987), M.Sc. (1990) and Ph.D. (1996) in Industrial Engineering from Gazi University. Her research concerns reliability optimization and modeling of complex systems using stochastic optimization techniques. She is a member of the Operations Research Society of Turkey.

Dr. Alice E. Smith; Department of Industrial Engineering; University of Pittsburgh; Pittsburgh, PA 15261 USA.

Internet (e-mail): aesmith@engrng.pitt.edu

Alice E. Smith is Associate Professor of Industrial Engineering and Board of Visitors Faculty Fellow at the University of Pittsburgh. Her research in modeling and optimization of

complex systems has been funded by Lockheed Martin Corporation, ABB Daimler Benz Transportation (ADtranz), the National Institute of Standards (NIST), the Ben Franklin Technology Center of Western Pennsylvania and the National Science Foundation (NSF), from which she was awarded a CAREER grant in 1995. She is a member of the Design and Manufacturing Editorial Board of *IIE Transactions* and is an associate editor of *INFORMS Journal on Computing, International Journal of Smart Engineering System Design,* and *Engineering Design and Automation.* She is a Senior Member of IIE, IEEE and SWE, a member of INFORMS and ASEE, and a Registered Professional Engineer in the Commonwealth of Pennsylvania.