IIE Transactions
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/uiie20

Economic design of reliable networks

DARREN L. DEETER [a] & ALICE E. SMITH [a]
[a] Department of Industrial Engineering , University of Pittsburgh , 1031 Benedum Hall, Pittsburgh, PA, 15261, USA E-mail:
Published online: 31 May 2007.

To cite this article: DARREN L. DEETER & ALICE E. SMITH (1998) Economic design of reliable networks, IIE Transactions, 30:12, 1161-1174, DOI: 10.1080/07408179808966573

To link to this article: http://dx.doi.org/10.1080/07408179808966573

IIE Transactions (1998) **30**, 1161–1174

Economic design of reliable networks

DARREN L. DEETER and ALICE E. SMITH

Department of Industrial Engineering, 1031 Benedum Hall, University of Pittsburgh, Pittsburgh, PA 15261, USA
E-mail: aesmith@engrng.pitt.edu

This paper describes a general approach to the optimal design of communications networks when considering both economics and reliability. The approach uses a genetic algorithm to identify the best topology of network arcs to collectively meet cost and network reliability considerations. This approach is distinct because it is highly flexible and can readily solve many versions of the network design problem, including formulations not previously seen in the literature that more closely reflect actual design scenarios. The method is shown to be effective, computationally efficient and flexible on a suite of diverse test problems.

1. Introduction

The problem of how to cost effectively design a network so that certain constraints are met and an objective is optimized is relevant in many real world applications such as telecommunications [1–3], computer networking [4–6], sewage systems [7], and oil and gas lines [7]. This paper focuses on the design of minimum cost reliable networks when a set of nodes and their topology are given, along with a set of possible bi-directional arcs that connect them. It is assumed 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. These are common assumptions among this family of problems [1–12]. Each possible arc has a known reliability and cost per unit distance. 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 [1,2,4,8,10,12]. 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.

This network design problem is an NP-hard combinatorial optimization problem [13] where the search space for a fully connected network with N nodes and k possible arc choices is:

$$k^{(|N|\times(|N|-1))/2}. \tag{1}$$

Compounding the exponential growth in the number of possible network architectures is the point that the calculation of network reliability is also an NP-hard problem, which grows exponentially with the number of arcs. Previous approaches have either been enumerative-based, which are applicable only for small network sizes [2,6,14], or heuristic-based which can be applied to larger networks but do not guarantee optimality.

Previous heuristic approaches include tabu search [9,15], genetic algorithms [7,8,11,12,16], simulated annealing [1,3,5] and others [4]. Kumar *et al.* have applied Genetic Algorithms (GA) to the problem of network design [11] and network expansion [12] when considering several objectives and constraints. The closest version of their research to this paper is network design when maximizing reliability given a cost constraint. Their GA approach to this problem 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 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 would tend to disrupt the beneficial effects of GA crossover. Finally, while the approach of Kumar *et al.* [11,12] might be expanded to consider alternative formulations, the papers address only all-terminal network reliability with arcs of identical reliability and unit cost.

In fact, 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 mathematical approaches to the problem, not of the de-

sign 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 imperative to allow designs with arcs of differing unit costs. For example, in telecommunications, a cable can be sheathed, or not. The research presented in this paper 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.

Another shortcoming of previous approaches is their limitation to a single network reliability metric and a single optimization formulation using cost and reliability. These are also relaxed in this paper, and a general, flexible design optimization approach is put forth. This approach is demonstrated on two reliability metrics (all-terminal and source-sink) and with two different objectives (maximizing reliability given a cost constraint and minimizing cost given a reliability constraint). The all-terminal reliability (also called the uniform or overall reliability) of a network is the probability that every node in the network design can communicate with every other node, over a specified mission time. The source-sink network reliability is the probability that the source node in the network can communicate with the sink node, also over a specified mission time.

This research uses a genetic algorithm because of its effectiveness and flexibility in solving many NP-hard combinatorial problems including those of reliability design [17–21]. The paper discusses the GA, network reliability calculation and the search strategy. The optimization approach is demonstrated on five test problems and is shown to be flexible, powerful and robust, and is computationally tractable for even large-scale networks.

2. Description of the approach

Genetic algorithms were first described by Holland in the 1970s and were inspired by natural selection [22]. In nature, the best members of a population survive and breed, thus letting the good traits of a species be passed on from one generation to the next. Spontaneous mutation results in new gene combinations that may be beneficial. Similarly in GA's the best members of a population of designs (*parents*) are blended to form new members (*children*) in the hope that the good design components are passed on to the next generation. Designs are also randomly perturbed (*mutants*) to introduce the concept of mutation. Eventually, through the repetition of the reproduction process (*generations*), the population will evolve until it generates optimal or near-optimal solutions. The roots for this expectation lie in schema theory described by Holland [22] and Goldberg [23]. DeJong [24] first applied GA to the field of optimization. The notation used throughout the rest of the paper is:

N = Set of given nodes.
L = Set of possible arcs.
l_{ij} = Option of each arc. $l_{ij} \in \{0, 1, 2, \ldots, k-1\}$.
$p(l_k)$ = Reliability of arc option.
$c(l_k)$ = Unit cost of arc option.
$\mathbf{x}$ = Architecture of network design.
$C(\mathbf{x})$ = Total cost of network design.
C_0 = Maximum cost constraint.
$R(\mathbf{x})$ = Reliability of network design.
R_0 = 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.

A genetic algorithm lends itself to the problem of economic design of reliable networks because each network design $\mathbf{x}$ is easily formed into a k-ery string (a string where each position can be one of k values) which can be used as a chromosome for the genetic algorithm. A *chromosome* is a GA term that is used to describe the encoded solution for a particular problem. Each place of the chromosome string is called an *allele*. Thus if the chromosome is of size ten then it contains ten alleles. Contained in the chromosome is all the information needed to distinguish a particular solution of the problem. In this case each allele of the chromosome represents a possible arc in the network design problem so there are $N(N-1)/2$ alleles in each chromosome architecture $\mathbf{x}$. The value of each of these elements tells which type of connection, l, the specific arc has with the pair of nodes, ij, adjacent to it.

The following example for a problem with $N = 5$ and $k = 4$ levels of connections shows how a candidate network design is encoded as a chromosome. Notice that the same solution is represented in both the arc matrix of Fig. 1 and in Table 1. There are $(5 \times 4)/2 = 10$ possible arcs for this example but only five are present; the other five are at level of connection $l_{ij} = 0$. This information is placed in a chromosome by copying and concatenating each row of the upper triangular of the matrix into the chromosome as is seen below (note that the only possible values allowed in each allele of the chromosome are $0, 1, \ldots, k-1$):

Chromosome: {0100203102}.

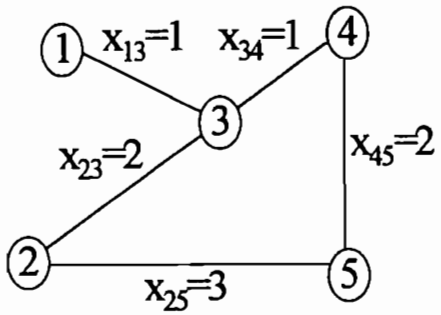

Fig. 1. Example network design.

In this paper, the objective is to find the minimum cost network architecture that meets a pre-specified network reliability:

$$\text{Minimize: } C(\mathbf{x}),$$
$$\text{subject to } R(\mathbf{x}) \geq R_0.$$

This formulation differs from the usual method of maximizing reliability subject to a maximum cost constraint as in Kumar *et al.* [11,12] and is computationally more difficult because the constraint is no longer linear. However, in network design the overriding constraint is reliability, which must meet a minimum level. Cost is dependent on first meeting the reliability constraint, and is therefore best handled as an objective rather than a hard constraint. If the network reliability is below the user-specified reliability, the solution is considered infeasible. Feasibility is especially hard to maintain in a GA because the blending and perturbation of solutions tends to disrupt feasibility. Rather than discard infeasible designs, it is better to penalize them according to their distance from feasibility and on the length of the optimization process [19]. This will encourage the optimization process to concentrate where the reliability constraint is active, which will tend to result in minimum cost feasible networks. The objective value above is therefore modified when the network does not meet the minimum reliability constraint to:

$$C_p(\mathbf{x}) = C(\mathbf{x}) + C(\mathbf{x}^*) \times (1 + R_0 - R(\mathbf{x}))^{r_p + (s \times g/50)}, \quad (2)$$

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.

Table 1. Five node connectivity matrix

	1	*2*	*3*	*4*	*5*
1	–	0	1	0	0
2	0	–	2	0	3
3	1	2	–	1	0
4	0	0	1	–	2
5	0	3	0	2	–

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

- ***Initialize parameters***
  - $g_{\max}$
  - s
  - $m\%$
  - r_m
  - r_p
  - reliability calculation method – backtracking or simulation
- ***Randomly generate initial population***, $g = 1$
  - Send initial population to the reliability calculation function
  - Send initial population to the cost calculation function
    - infeasible members are penalized
  - 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 in new population
    - two distinct parents are chosen using a rank-based procedure
    - children are generated using uniform crossover
    - after a child is created it is mutated
    - when enough children are created the parents are replaced by the children
  - Send new population to the reliability calculation function
  - Send new population to the cost calculation function
    - infeasible members are penalized
  - Check for New Best Solution
    - if no solution is feasible the best infeasible solution is recorded
  - Repeat until $g = g_{\max}$

2.1. *Selection and breeding*

Parents are selected to breed using a rank based quadratic procedure [25]. All the parents in the population are ordered according to their level of fitness with 1 being the highest level of fitness, then a random number between 0 and $\sqrt{s}$ is generated. The random number is squared, truncated and 1 is added to it. The resulting number is the rank of the parent chosen. A parent is not allowed to breed with itself.

Blending is done using uniform crossover [23]. This is accomplished 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 population is created it goes through a mutation process. A solution undergoes mutation according to the percentage of population mutated. For example if $m\% = 20\%$ and $s = 30$, then six 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, $l_{ij} = 0$, 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}

2.2. *Calculation of network reliability*

The problem of calculating or estimating the reliability of a network is another active area of research related to the economic network design problem. There are two main approaches – exact calculation through analytic methods [10,26–28] and estimation through variations of Monte Carlo simulation [29–36]. For the all-terminal network reliability problem, efficient simulation is difficult because these methods generally lose efficiency as a network approaches a fully connected state. There are also upper and lower bound expressions for network reliability [10,31,37], however these are too loose to be effective surrogates in the all-terminal design process. Furthermore, many bounding procedures and improved efficiency simulations depend on the assumption that all arcs have the same reliability, which is relaxed in this research. Therefore in this paper, either the network reliability was calculated exactly using a backtracking procedure or a classic Monte Carlo procedure estimated the network reliability.

The backtracking algorithm from Ball and Van Slyke [26] is given below and exactly calculates the network unreliability ($1 - R(\mathbf{x})$).

Step 0. (Initialization). Mark all arcs as free; create a stack that is initially empty.
Step 1. (Generate modified cutset)
(a) Find a set of free arcs that together with all inoperative arcs will form a network-cut.
(b) Mark all the arcs found in 1(a) inoperative and add them to the stack.
(c) The stack now represents a modified cutset; add its probability to a cumulative sum.
Step 2. (Backtrack)
(a) If the stack is empty, end.
(b) Take an arc off the top of the stack.
(c) If the arc is inoperative and if when made operative, a spanning tree of operative arcs exists, then mark it free and go to 2(a).
(d) If the arc is inoperative and the condition tested in 2(c) does not hold, then mark it operative, put it back on the stack and go to *Step 1*.
(e) If the arc is operative, then mark it free and go to 2(a).

Note that the algorithm above is for all-terminal reliability and needs to be modified for use in a source-sink design problem as below:

Step 0. (Initialization). Mark all arcs as free; create a stack that is initially empty.
Step 1. (Generate modified cutset)
(a) Find a set of free arcs that together with all inoperative arcs will form a source-sink cut.
(b) Mark all the arcs found in 1(a) inoperative and add them to the stack.
(c) The stack now represents a modified cutset; add its probability to a cumulative sum.
Step 2. (Backtrack)
(a) If the stack is empty, end.
(b) Take an arc off the top of the stack.
(c) If the arc is inoperative and if when made operative, a path from the source to the sink exists, then mark it free and go to 2(a).
(d) If the arc is inoperative and the condition tested in 2(c) does not hold, then mark it operative, put it back on the stack and go to Step 1.
(e) If the arc is operative, then mark it free and go to 2(a).

For larger networks, Monte Carlo simulation is used to accurately estimate network reliability. The network is simulated t times given the design and the arc reliabilities:

Initialize $i = 0, c = 0$.
Step 0. While $i < t$, Repeat.
Step 1. Randomly generate network.
(a) $i = i + 1$.
Step 2. Check to see if the network forms a spanning tree.
(a) If the network forms a spanning tree then $c = c + 1$, go to *Step 0*.
(b) If the network does not form a spanning tree go to *Step 0*.
Step 3. $R(\mathbf{x}) = c/t$.

When the need to simulate a network's reliability arises, other issues become important. One of these issues is whether or not the estimator is biased. The other issue is the variance of the estimate. Every Monte Carlo technique referenced is an unbiased estimator. The variance of the Monte Carlo method described above is:

$$\mathrm{Var}(R(\mathbf{x})) = \frac{R(\mathbf{x})(1 - R(\mathbf{x}))}{t}. \quad (3)$$

To get a more accurate reliability estimate, t must increase.

3. Test problems and results

Five test problems were studied and each exhibits a different aspect of the GA metaheuristic approach. The first problem is a five node network taken from Jan *et al.* [2] but expanded by changing the arcs from a simple on/off state to one of four possible states. These four arc types, shown in Table 2 were used for the first four test problems. Problem 1 is considered with seven different system reliability constraints and uses the backtracking algorithm to calculate $R(\mathbf{x})$ exactly. The second test problem is used to show the scale-up of this approach. It was generated by randomly choosing ten nodes on a 100 by 100 grid. Monte Carlo simulation was used to estimate reliability for this larger problem. The third test problem shows the flexibility of the approach by considering economic design of an 18 arc source-sink network. The fourth test problem is a 14 node problem that alters the objective function/constraint set by maximizing reliability given a maximum cost. Finally, the approach is demonstrated on a network design problem currently facing the government of Turkey. This is the connection of 19 university and research center sites located in nine cities by a high-speed data network.

While CPU time comparisons are difficult to make precisely because of the differences between hardware and software platforms and the issue of non-optimized code, Table 3 gives the average CPU times for each solution evaluated on a UNIX mainframe computer for each of the five problem sizes. Most of the CPU time is directed at calculating or estimating the network reliability (which forms part of the objective function), with much less time needed for the GA operators of crossover, mutation and selection. The results in the subsequent sections will show that the GA only examines a very small fraction of the possible search space, so these average CPU times taken with the number of solutions considered gives a reasonable appraisal of the computational effort involved in the process. It might also be noted that for design problems of this type, the issue of CPU time is not critical. These are not activities that must be performed in real time, and the user will typically have considerable flexibility in the computational effort allowed. Of course, the CPU effort must be within practical limits, thus precluding the use of exact algorithms for network reliability on large problems.

Table 2. Arc unit costs and corresponding reliabilities

Connection type (k)	*Reliability*	*Unit cost*
Not connected, 0	0	0
1	0.70	8
2	0.80	10
3	0.90	14

Table 3. CPU effort per solution evaluated for different problem sizes

Problem size	*CPU seconds per solution*
5 nodes	0.0066
10 nodes	0.1958
18 arcs, source-sink	0.0265
14 nodes	0.6783
19 nodes	1.5023

3.1. *Test problem 1 – five nodes*

The problem was created by using the test problem arc costs of Jan *et al.* [2] as distances and using the unit costs and reliabilities shown in Table 2. Since this problem has $k = 4$ levels its search space size is $4^{(5\times4)/2} = 1048576$. This problem was considered at seven different system reliability levels seen in Table 4 to examine performance at different levels of constraint. Since this was a relatively small problem it was possible to enumerate the exact solutions for the seven different system reliability levels using the backtracking algorithm [26]. The optimal solutions for each level of constraint appear with the solution vectors $\mathbf{x}$ in Table 4. Note each solution vector $\mathbf{x}$ makes use of two or more arc connection levels clearly showing the value of expanding the search space to consider different cost/reliability levels. Furthermore, some of the designs are counterintuitive and would probably not be identified by human designers. For ex-

Table 4. Optimal solutions for test problem 1

R_0	$C(\mathbf{x})$	$R(\mathbf{x})$	$\mathbf{x}$
0.99900	5522	0.99908	3323333323
0.99500	4352	0.99518	3113311313
0.99000	3754	0.99052	3203322303
0.95000	2634	0.95353	3003302303
0.93125	2416	0.93361	2003301303
0.90000	2184	0.91854	3003300303
0.85000	1904	0.85536	3003200203

ample, the second most constrained version where $R(\mathbf{x})$ must be 0.995 or above includes many of the least reliable arc type ($l_{ij} = 1$).

After exploratory runs of the GA, the following were set: $s = 40$, $m\% = 25$, $r_m = 0.25$, $r_p = 6$, and $g_{\max} = 6000$. Each level of constraint was run over the same set of ten different random number seeds. Because GA is a stochastic algorithm, it is important to characterize the variability due to the choice of random seed. In the most constrained versions of the problem ($R_0 = 0.999$ or 0.995) the GA found the optimal solution in many of the runs, and always converged to a feasible, near-optimal solution. For the other five versions ($R_0 = 0.99, 0.95, 0.93125, 0.90, 0.85$) the optimal solution was found using each of the ten seeds. Overall, the optimal solution is found in fewer total evaluations as the severity of the constraint of the problem is relaxed, i.e., as R_0 is lowered. As is seen in the summary in Table 5, the average portion of the solution space that must be searched in order to find the optimal solution is fairly consistent. Note that this is an upper bound as there is no accounting for duplicate solutions that are likely to occur in a GA search. Considering the proportion of the possible network designs examined (0.44–11.34%), that optimal solutions were obtained in most cases, and very good, feasible solutions were obtained otherwise, the GA appears to be very effective. Conserving the proportion of the solution space searched is a primary concern in network reliability problems where the calculation or estimation of reliability is computationally expensive.

3.2. *Test problem 2 – 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 (Appendix, Table A1) and the unit costs and reliabilities were from Table 2. 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 using backtracking. 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 in the estimator was dynamic. At the first generation, the estimator replicated each system 1000 times ($t = 1000$). 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 ($t = t + 1000$). This dynamic approach was used so that as the search progressed the reliability estimates would get better. Whenever a solution 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 $t = 25000$ times. The best check was used to help ensure the feasibility and accuracy of the very best candidate designs.

From initial experimentation $s = 90$ and $g_{\max} = 1200$, with the other parameters remaining the same as in test problem 1. The search space and the chromosome length are both much larger than in the previous test problem, prompting the increase in the number of designs searched. Since the ten node 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. In the greedy search, the same number of designs were selected and the cost of every design was calculated. If the cost of a particular design was less than the cost of the best design found so far then the reliability of that solution was calculated using the best check reliability estimator described above. If the estimated reliability was greater than R_0 then the new solution became the new best solution. Ten runs of each algorithm using the same set of random number seeds were averaged and plotted as shown in Fig. 2.

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

Table 5. Summary results of test problem 1

R_0	*Mean solutions searched*	*Percentage of space searched*	*Number optimal*
0.99900	40560	3.87	8 of 10
0.99500	23200	2.22	4 of 10
0.99000	31400	2.99	10 of 10
0.95000	118880	11.34	10 of 10
0.93125	25560	2.44	10 of 10
0.90000	26160	2.49	10 of 10
0.85000	4640	0.44	10 of 10

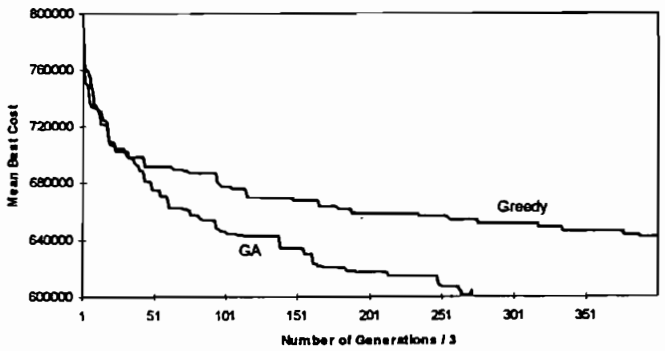

Fig. 2. GA versus greedy search averaged over ten runs.

Table 6. Test problem 2 results over ten seeds

Seed	*Number of evaluations*	*Percentage of space searched*	*R*(x)	*95% CI of R*(x)	*Min C*(x)
1	22400	0.181E-24	0.9544	(0.952, 0.957)	615664
2	24400	0.197E-24	0.9568	(0.954, 0.960)	604292
3	8000	0.646E-24	0.9616	(0.959, 0.964)	588142
4	23600	0.191E-24	0.9520	(0.949, 0.955)	566132
5	35600	0.288E-24	0.9524	(0.950, 0.955)	639494
6	26000	0.210E-24	0.9516	(0.949, 0.954)	626620
7	42800	0.346E-24	0.9504	(0.948, 0.953)	596781
8	38400	0.310E-24	0.9508	(0.948, 0.954)	579022
9	31200	0.252E-24	0.9560	(0.953, 0.959)	606448
10	29200	0.236E-24	0.9616	(0.959, 0.964)	610862

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.

The final GA results appear in Table 6. Since this problem was so large that a reliability estimation had to be used, a confidence interval for the reliabilities also appears in Table 6. This confidence interval was calculated using the variance estimator in Equation 3. Note that six of the ten solutions have a 95% confidence interval that lies completely above the 0.95 reliability constraint.

Figure 3 is the best cost design from the ten runs (cost = 566132, reliability = 0.952). The nodes are marked with a diamond and the level of the each arc is represented by the number near it. Note that all three levels of arc were used in this design indicating that the restrictive assumption of a single arc reliability and cost does not result in correct economic design.

3.3. *Test problem 3 – source-sink*

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 is shown in Fig. 4. The GA easily accommodates these rather fundamental changes. The change in the reliability calculation is accomplished by simply modifying the backtracking algorithm as was described in Section 2.2. The fact that not all possible arcs are allowed in this design is accommodated by simply leaving these arcs out of the chromosome string.

This problem is taken from the literature [34,35] and has 6.9×10^{10} possible architectures, thus precluding enumeration to identify the optimal design. The distance matrix for this problem appears in the Appendix as Table A2. A system reliability requirement $R_0(\mathbf{x}) = 0.99$ is set. After some initial experimentation it was determined that $s = 40$, $m\% = 80$, $r_m = 0.05$ and $g_{\max} = 2000$. The remaining settings were as in the previous problems.

Results are presented in Table 7. Seven of the ten runs found a best cost of 4680 with an architecture as shown in Fig. 5, or a symmetric copy. The other three test runs found a best cost of 4726 and had the architecture shown in Fig. 6, or a symmetric copy. Since the GA found only two distinct solutions over ten runs, it is likely that both are near-optimal, if 4680 is not optimal. The results of comparison with the greedy search algorithm described for test problem 2 appear in Fig. 7. Note that the GA not only converges to a superior solution, but also converges at a greater speed, evidencing both effectiveness and efficiency.

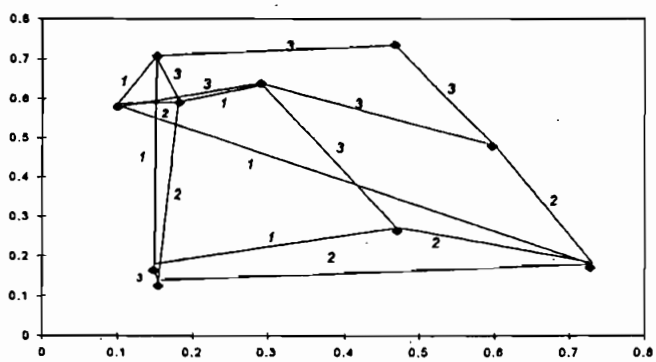

Fig. 3. Best solution for randomly generated ten node layout.

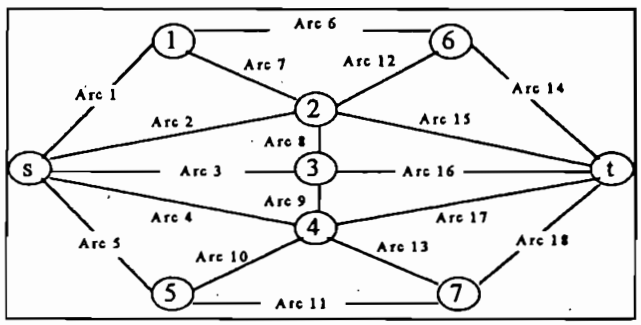

Fig. 4. Source-sink problem topology.

Table 7. Test problem 3 results over ten seeds

Seed	*Number of evaluations*	*Percentage of space searched*	*Solution*
1	4000	0.582E-7	Fig. 6
2	32400	0.471E-6	Fig. 6
3	65600	0.955E-6	Fig. 6
4	60400	0.879E-6	Fig. 5
5	6400	0.931E-7	Fig. 5
6	45600	0.664E-6	Fig. 5
7	12400	0.180E-6	Fig. 5
8	2400	0.349E-7	Fig. 5
9	72400	0.105E-5	Fig. 5
10	43600	0.634E-6	Fig. 5

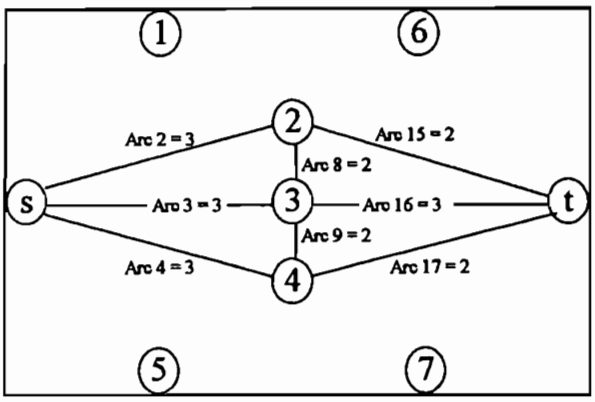

Fig. 6. Source-sink solution with cost = 4726 and $R(\mathbf{x}) = 0.9902$.

3.4. *Test problem 4 – 14 City LAN*

Demonstrating this problem on a realistic, scaled-up application highlights the applicability of this work. This 14 node problem was constructed by selecting cities in the US and computing the Euclidean distances between them using their coordinates (Appendix, Table A3) and using the four arc choices from Table 2. The search space of this problem is 6×10^{54}. Besides the scale-up issue, the difference between this problem and the other design problems is that this problem illustrates the flexibility of the GA by reversing the constraint and the objective function. In this problem the objective is to maximize system reliability, subject to a cost constraint, prompting some adjustments to the penalty function. Since the cost now forms the constraint, the penalty was applied to the reliability of infeasible solutions (when $C(\mathbf{x}) > C_0$) using Equation 4. Note that the primary objective of the penalty function is the same, solutions are penalized to the degree that they violate the cost constraint. C_0 is the maximum cost allowed ($C_0 = 22000000$) and $R_{\text{BEST}}(\mathbf{x})$ is the maximum reliability feasible solution. This means any infeasible solution will be inferior to at least the best feasible solution, and in the latter phases of the search when the system reliability nears 1.0, this penalty becomes a "death penalty." That is, only feasible solutions are maintained in the population. A discussion of the death penalty applied to constrained reliability optimization problems can be found in Coit and Smith [18]. The penalty function of this problem is much less critical than that of test problems 1 through 3 because for this formulation, many feasible (low cost) solutions can be readily identified. When reliability is the constraint, it is not always easy to identify even a single feasible solution.

$$R(\mathbf{x}) = R_{\text{BEST}}(\mathbf{x}) - \frac{(C(\mathbf{x}) - C_0)}{C_0}. \tag{4}$$

For this problem, the Monte Carlo iterations were increased since $R(\mathbf{x})$ is the objective. For the general reliability check, which was used on every new population member, $t = 2500$ replications. Then for the best check, which was used to verify that a potential new best solution did have a reliability better than the current best solution it was about to replace, $t = 50000$ repetitions. The best check used a very large t value because as each generation gets better, the reliability of the best solution approaches 1.0, thus requiring a more accurate reliability estimate. This GA was applied using $s = 90$, $m\% = 25$, $r_m = 0.25$, and $g_{\max} = 250$.

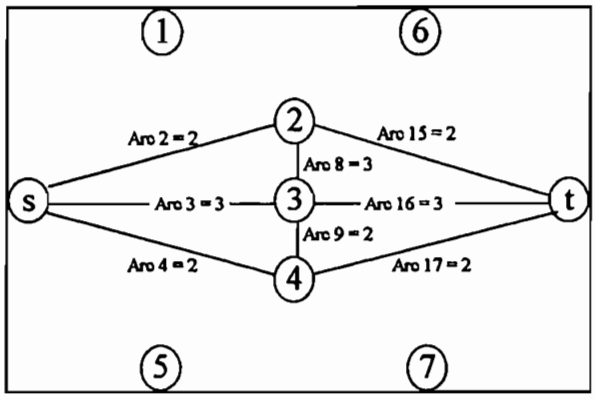

Fig. 5. Source-sink solution with cost = 4680 and $R(\mathbf{x}) = 0.99000$.

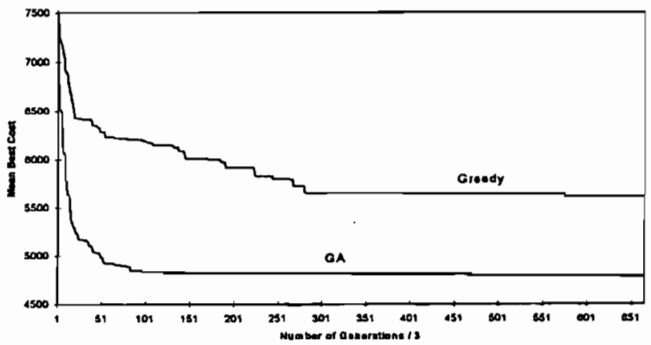

Fig. 7. Source-sink GA versus greedy search.

Table 8. Fourteen node LAN problem results over five runs

Seed	*C*(x)	*Best R*(x)	*95% CI of R*(x)
1	21593632	0.9954	(0.9948, 0.9960)
2	21803616	0.9972	(0.9967, 0.9977)
3	21656910	0.9981	(0.9977, 0.9985)
4	21918120	0.9972	(0.9967, 0.9977)
5	20931202	0.9965	(0.9960, 0.9970)

Five runs were made using different random seeds and the results are shown in Table 8. The topology of the best solution (from random seed 3) is presented in the Appendix as Table A4. Since the computational effort required was burdensome, no reasonable comparisons could be made. The solutions are all feasible and highly reliable with similar system reliabilities. More importantly, this problem demonstrates that the GA method can work on a very large problem and with a reversed objective function. Furthermore, since the GA is an iterative search, for large problems, the search may be terminated at any time and still return a feasible and very good network design.

3.5. *Test problem 5 – Turkish universities network*

This problem is a simplified version of a real network design problem currently facing the federal government of Turkey. Turkey is a rapidly developing country in Asia Minor with a population of 64 million. The Turkish government views communications infrastructure as a main factor to boost economic development and has initiated large communications projects since the 1980s. As a result, Turkey has the newest and largest communications network in Eastern Europe and the Middle East. Because of continued growth in internet use, the Scientific and Technological Research Council of Turkey (Tubitak) established ULAK-NET, a high speed data network linking universities and research centers. ULAK-NET will be expanded to include fiber-optic links between 19 academic centers in Turkey located in nine cities throughout the country. The node neighborhoods, their cities and the distance matrix in kilometers can be found in the Appendix as Table A5. Unit costs for fiber optic cable at three reliabilities were ascertained. Node costs (fiber optic terminals) were not considered here. The cable unit costs included material costs, splicing costs, repeater costs and placement costs (trenching, boring, etc.). The cable reliabilities and the unit costs were: (0.960, 333), (0.975, 433) and (0.990, 583). The reliability requirement for the network was set at 0.99, and the objective was to minimize cost.

The genetic algorithm was run as in Test Problem 2 with an increase of r_m to 35% and an increase in Monte Carlo replications to 9000 (at the initial generation) for the general check and 120000 for the best check. The

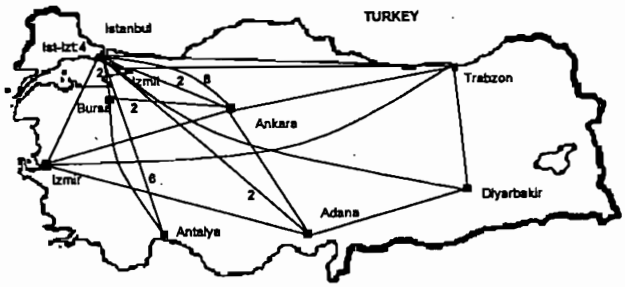

Fig. 8. Best network to connect 19 universities in nine Turkish cities.

increase in simulation replications was necessary because of the precision needed for a real design problem. This did cause increase in computational effort as reflected in Table 3. The best solution that has a lower confidence limit of reliability of greater than 0.99 after 400 generations is shown in Fig. 8. The numbers of connections between cities, if greater than one, are shown in the bold numbers. The exact architecture of this solution is shown in the Appendix as Table A6. This solution has a cost of 7694708, an expected reliability of 0.99967 and a 95% confidence interval about reliability of (0.99401, 1.00000).

4. Conclusions

Communications and data networks are becoming increasingly important, ranging from small networks within a building or plant to global networks for high-speed data transmission. These networks must all meet reliability criteria, usually in the form of a minimum network reliability metric. Economics also plays a critical role as these networks will incur large first costs during their construction. Furthermore, design problems are usually accomplished iteratively, where a crude design is refined as more information and criteria are identified. Therefore, a flexible, general approach is more useful than narrow and restrictive special purpose algorithms that solve only particular versions of the design problem or do not scale-up to realistically sized problems. This paper has considered the general problem of economic design of reliable networks using an evolutionary approach that cannot guarantee optimality, but has been demonstrated to be effective on a wide range of problems.

If a user can be assured of optimal, or near-optimal, results when exerting a small fraction of the computational effort required for enumerative methods there is strong motivation to use a heuristic, such as the one developed in this paper. There is an added attraction that since the GA is an iterative technique, which generally achieves diminishing improvements in objective function value as the search continues, the user may terminate the search at any time and still have a very good, feasible

solution. Furthermore, not only a single good, feasible solution is returned. The user may examine a group of superior solutions if further investigation is warranted. Besides the effectiveness and computational practicality of GA, a strong attraction is the flexibility of the method. With only minor alterations in the algorithm code, problems with differing objectives and constraints, alternative methods of calculating or estimating network reliability, and varying restrictions on arc topologies can be addressed. The approach could be readily expanded to handle additional design considerations besides cost and reliability, such as speed or volume. Similarly, expanding the design problem to consider node reliability and redundant arcs could be done. In summary, this framework is quite general and should be applicable to a wide range of network design problems where economics and reliability are considerations.

Acknowledgment

The authors very much appreciate the assistance of Abdullah Konak, who provided the data on the Turkish test problem. Alice E. Smith gratefully acknowledges financial support from the National Science Foundation CAREER grant DMI-9502134.

References

[1] Atiqullah, M.M. and Rao, S.S. (1993) Reliability optimization of communication networks using simulated annealing. *Microelectronics and Reliability*, **33**, 1303–1319.

[2] Jan, R.-H., Hwang, F.-J. and Chen, S.-T. (1993) Topological optimization of a communication network subject to a reliability constraint. *IEEE Transactions on Reliability*, **42**, 63–70.

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

[4] Aggarwal, K.K., Chopra, Y.C. and Bajwa, J.S. (1982) Topological layout of links for optimising the overall reliability in a computer communication system. *Microelectronics and Reliability*, **22**, 347–351.

[5] Fetterolf, P.C. and Anandalingam, G. (1992) Optimal design of LAN-WAN internetworks: an approach using simulated annealing. *Annals of Operations Research*, **36**, 275–298.

[6] Wilkov, R.S. (1972) Design of computer networks based on a new reliability measure, in *Proceedings of the Symposium on Computer-Communications Networks and Teletraffic*, Fox, I. (ed.), Polytechnic Institute of Brooklyn, Brooklyn, NY. pp. 371–384.

[7] Walters, G.A. and Smith, D.K. (1995) Evolutionary design algorithm for optimal layout of tree networks. *Engineering Optimization*, **24**, 261–281.

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

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

[10] Jan R.-H. (1993) Design of reliable networks. *Computers and Operations Research*, **20**, 25–34.

[11] Kumar, A., Pthak R.M. Gupta, Y.P. and Parsaei, H.R. (1995) A genetic algorithm for distributed system topology design. *Computers and Industrial Engineering*, **28**, 659–670.

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

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

[14] Aggarwal, K.K. and Rai, S. (1981) Reliability evaluation in computer-communication networks. *IEEE Transactions on Reliability*, **R-30**, 32–35.

[15] Koh, S.J. and Lee, C.Y. (1995) A tabu search for the survivable fiber optic communication network design. *Computers and Industrial Engineering*, **28**, 689–700.

[16] Deeter, D.L. and Smith, A.E. (1997) Heuristic optimization of network design considering all terminal reliability, in *Proceedings of the Reliability and Maintainability Symposium*, IEEE, Piscataway, NJ. pp. 194–199.

[17] Coit, D.W. and Smith, A.E. (1996) Reliability optimization of series-parallel systems using a genetic algorithm. *IEEE Transactions on Reliability*, **45**, 254–260.

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

[19] Coit, D.W., Smith, A.E. and Tate, D.M. (1996) Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing*, **8**, 173–182.

[20] Ida, K., Gen M. and Yokota, T. (1994) System reliability optimization with several failure modes by genetic algorithm, in *Proceedings of 16th International Conference on Computers and Industrial Engineering*, Gen, M. and Yamazaki, G. (eds), Pergamon Press, New York. pp. 349–352.

[21] Painton, L. and Campbell, J. (1995) Genetic algorithms in optimization of system reliability. *IEEE Transactions on Reliability*, **44**, 172–178.

[22] Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

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

[24] DeJong, K.A. (1975) An analysis of the behaviour of a class of genetic adaptive systems. Ph.D. thesis, University of Michigan, Ann Arbor, MI.

[25] Tate, D.M. and Smith, A.E. (1995) A genetic approach to the quadratic assignment problem. *Computers and Operations Research*, **22**, 73–83.

[26] Ball, M. and Van Slyke, R.M. (1977) Backtracking algorithms for network reliability analysis. *Annals of Discrete Mathematics*, **1**, 49–64.

[27] Hanzhong, C. and Dongkui, L. (1994) A new algorithm for computing the reliability of complex networks by the cut method. *Microelectronics and Reliability*, **34**, 175–177.

[28] Mandaltsis, D. and Kontoleon, J.M. (1987) Overall reliability determination of computer networks with hierarchical routing strategies. *Microelectronics and Reliability*, **27**, 129–143.

[29] Cancela, H. and El Khadiri, M. (1995) A recursive variance- reduction algorithm for estimating communication-network reliability. *IEEE Transactions on Reliability*, **44**, 595–602.

[30] Easton, M.C. and Wong, C.K. (1980) Sequential destruction method for Monte Carlo evaluation of system reliability. *IEEE Transactions on Reliability*, **R-29**, 27–32.

[31] Fishman, G.S. (1986) A Monte Carlo sampling plan for estimating network reliability. *Operations Research*, **34**, 581–594.

[32] Fishman, G.S. (1986) A comparison of four Monte Carlo methods for estimating the probability of $s - t$ connectedness. *IEEE Transactions on Reliability*, **R-35**, 145–155.

[33] Kamat, J.S. and Riley M.W. (1975) Determination of reliability using event-based Monte Carlo simulation. *IEEE Transactions on Reliability*, **R-4**, 73–75.

[34] Kumamoto, T., Tanaka, K. and Inoue, K. (1977) Efficient evaluation of system reliability by Monte Carlo method. *IEEE Transactions on Reliability*, **R-26**, 311–315.

[35] Mazumdar, M., Coit, D.W. and Shih, F.-R. (1995) An efficient Monte Carlo method for assessment of system reliability based on a Markov model. Technical report, Department of Industrial Engineering, University of Pittsburgh.

[36] Yeh, M.-S., Lin, J.-S. and Yeh, W.-C. (1994) A new Monte Carlo method for estimating network reliability, in *Proceedings of the 16th International Conference on Computers and Industrial Engineering*, Gen, M. and Yamazaki, G. (eds), Pergamon Press, New York. pp. 723–726.

[37] Shinmori, S., Koide, T. and Ishii, H. (1995) On lower bound for network reliability by edge-packing. *Transactions of the Japan Society for Industrial and Applied Mathematics*, **5**, 139–151.

Appendix

Table A1. Distance matrix for test problem 2

	2	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	47.3995	41.7519	24.9990	48.8760	54.7605	33.7527	34.6182	43.7072	27.3028
2		20.3195	28.9359	39.9203	31.7823	65.4767	68.7027	32.1302	62.2745
3			34.7425	19.8154	15.4418	49.4191	53.0689	11.8351	64.1167
4				50.8140	50.1819	54.7977	56.7332	43.0894	33.3708
5					13.6208	41.8359	45.8073	8.2105	75.0041
6						54.1884	58.1259	12.0843	78.7647
7							3.9718	42.6274	58.0398
8								46.5197	57.6986
9									68.8805

Table A2. Distance matrix for test problem 3

	1	*2*	*3*	*4*	*5*	*6*	*7*	*t*
s	58	63	60	63	58	–	–	–
1		42	–	–	–	60	–	–
2			20	–	–	42	–	63
3				20	–	–	–	60
4					42	–	42	63
5						–	60	–
6							–	58
7								58

Table A3. Distance matrix for test problem 4

	2	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*
1	57697	94650	59759	37381	55977	68070	33246	42618	75620	67259	52730	48346	54074
2		36953	61304	42911	31288	96035	7860	99950	17924	9627	21077	73638	3650
3			85833	74960	55607	125096	113733	100000	19032	27431	49422	104348	40579
4				25503	31131	40634	48556	83036	72047	67534	40387	25588	59203
5					22590	53316	40233	70044	57833	51240	25566	30733	39923
6						69492	62712	92232	40935	36569	10465	49111	29885
7							37859	6748	109671	103864	76816	22949	93172
8								34827	95562	87998	65112	25500	75146
9									117844	109566	91856	60199	96304
10										8434	32875	88055	21556
11											27341	81812	13277
12												55242	19459
13													70650

Table A4. Best solution found for test problem 4; $R(\mathbf{x}) = 0.9981$ and $C(\mathbf{x}) = 21656910$

	2	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*
1		1	1	3	3	2				2		2	1
2		1	1	1			1		2	2	3		1
3				1				2		2		1	
4							2			1		1	2
5						3		1				2	
6									2	1	2	2	2
7								2	1			2	
8								3	1				1
9											1		1
10										3	3		
11													1
12												2	
13													

1. Atlanta
2. Baltimore
3. Boston
4. Chicago
5. Cincinnati
6. Cleveland
7. Kansas City
8. Memphis
9. New Orleans
10. New York
11. Philadelphia
12. Pittsburgh
13. St. Louis
14. Washington DC

Table A5. Distance matrix for test problem 5

Research center	*Tahttak*	*Atakoy*	*Levent*	*Ayazaga*	*Gay 1*	*Gay 2*	*Bursa*	*Yeniseh*	*Trabzon*	*Ulus 1*	*Ulus 2*	*Acib 1*	*Acib 2*	*Kadikoy*	*Izmir*	*Anttalya*	*Adana*	*Diyar-bakir*
Izmit	111	126	120	122	115	116	132	346	968	343	344	106	107	105	454	613	828	1261
Tahtakale*	–	15	15	17	5	6	243	458	1079	454	456	10	11	5	565	724	939	1342
Atakoysup >		–	15	17	13	14	258	473	1094	469	471	25	26	23	580	740	954	1357
Levent*			–	2	5	6	248	460	1082	456	457	12	13	15	570	730	943	1353
Ayazaga*				–	8	9	251	463	1085	459	460	15	16	18	573	733	946	1355
Gayrettepe 1*					–	1	246	457	1080	454	455	10	9	12	568	728	940	1350
Gayrettepe 2*						–	245	456	1079	453	454	9	8	11	567	727	939	1351
Bursa							–	384	383	380	381	235	236	240	322	542	831	1301
Yenisehir**								–	766	3	4	450	451	453	580	542	487	920
Trabzon									–	763	764	1074	1075	1077	1345	1307	972	624
Ulus 1**										–	1	450	451	453	582	544	489	921
Ulus 2**											–	449	450	452	583	545	490	922
Acibadem 1*												–	1	4	560	720	932	1337
Acibadem 2*													–	3	561	721	933	1338
Kadikoy*														–	563	723	934	1340
Izmir															–	469	898	1424
Antalya																–	553	1079
Adana																	–	526
Diyarbakir																		–

* In Istanbul, Turkey
** In Ankara, Turkey

Table A6. Best solution for problem 6

Research center	*Tahttak*	*Atakoy*	*Levent*	*Ayaazaga*	*Gay 1*	*Gay 2*	*Bursa*	*Yeniseh*	*Trabzon*	*Ulus 1*	*Ulus 2*	*Acib 1*	*Acib 2*	*Kadikoy*	*Izmir*	*Anttalya*	*Adana*	*Diyar-bakir*
Izmit		1	1					2	2	1		1		2				
Tahtakale*			1				1									2		
Atakoy*			2		3			2					3			1		
Levent*											3							
Ayazaga*					2				2		2							
Gayrettepe 1*						1		1								2		2
Gayrettepe 2*							1			2				3				
Bursa								1			1					2		
Yenisehir**											2							
Trabzon															1			1
Ulus 1**											3	2	1	1			3	
Ulus 2**																		
Acibadem 1*															1	1		
Acibadem 2*														1		1	1	
Kadikoy*																2		
Izmir																	1	
Antalya																		
Adana																		1
Diyarbakir																		

* In Istanbul, Turkey
** In Ankara, Turkey

Biographies

Darren Deeter was a graduate student in the Industrial Engineering Department of the University of Pittsburgh and received an M.S.I.E. in the Summer of 1996. He currently works for Andersen Consulting in Chicago as an Analyst. His Master's Thesis focuses on heuristic combinatorial optimization of network design when considering reliability and cost. He has a B.S. in Mathematics and Physics from Allegheny College (Phi Beta Kappa) and an M.A. in Mathematics from Kent State University. He is a member of IIE, INFORMS and APICS.

Alice Smith is an Associate Professor of Industrial Engineering. After industrial experience with Southwestern Bell Corporation, she joined the faculty of the University of Pittsburgh in 1991. Her research interests are in modelling and optimization of complex systems using computational intelligence techniques, and her research has been sponsored by Lockheed Martin Corporation, ABB Daimler-Benz Transportation, the Ben Franklin Technology Center of Western Pennsylvania, the National Institute of Standards and the National Science Foundation, from which she was awarded a CAREER grant in 1995. Her work in design optimization when considering system reliability has appeared in *IEEE Transactions on Reliability, INFORMS Journal on Computing, IEEE Transactions on Evolutionary Computation, Computers and Operations Research* and *Computers and Industrial Engineering*. Dr. Smith is a senior member of IEEE, IIE and SWE, and a member of ASEE and INFORMS, and serves as an Associate Editor of *INFORMS Journal on Computing, IEEE Transactions on Evolutionary Computation, IIE Transactions* (Design and Manufacturing) and *Engineering Design and Automation*.