Reichelt, Dirk ; Rothlauf, Franz ; Gmilkowsky, Peter :

Designing reliable communication networks wih a genetic algorithm using a repair heuristic

Zuerst erschienen in:
Evolutionary computation in combinatorial otpimization. - Berlin [u.a.] : Springer, 2004
S. 177-187
(Lecture Notes in Computer Science ; 3004)

Designing Reliable Communication Networks with a Genetic Algorithm Using a Repair Heuristic

Dirk Reichelt[1], Franz Rothlauf[2], and Peter Gmilkowsky[1]

[1] Institute of Information Systems, Ilmenau Technical University
Helmholtzplatz 3, P.O. Box 100565, 98684 Ilmenau, Germany,
{Dirk.Reichelt@tu-ilmenau.de,Peter.Gmilkowsky}@tu-ilmenau.de
[2] University of Mannheim, Department of Information Systems I
Schloss, 68131 Mannheim, Germany
rothlauf@uni-mannheim.de

Abstract. This paper investigates GA approaches for solving the reliable communication network design problem. For solving this problem a network with minimum cost must be found that satisfies a given network reliability constraint. To consider the additional reliability constraint different approaches are possible. We show that existing approaches using penalty functions can result in invalid solutions and are therefore not appropriate for solving this problem. To overcome these problems we present a repair heuristic, which is based on the number of spanning trees in a network. This heuristic always generates a valid solution, which when compared to a greedy cheapest repair heuristic shows that the new approach finds better solutions with less computational effort.

1 Introduction

The optimal design of reliable communication and transportation networks is important in many application fields such as gas pipelines, communication networks, and electricity distribution. When designing reliable communication networks there is a trade-off between the necessary investments in the network and the quality of service provided to the network users. An important service measurement is the all-terminal reliability of the network which is defined as the probability that the network is still connected even if some nodes or links fail [1]. In the reliable communication network problem [2] communication links must be chosen such that the network costs are minimized given a network reliability constraint. Both the network design problem and the calculation of the network reliability, have been proven to be NP-hard [3,4]. Genetic Algorithms (GA) have shown promising results when applied to this problem [5,6,7].

In this paper we investigate existing GA approaches for the design of reliable communication networks and propose a heuristic that repairs each candidate solution with respect to the number of spanning trees in the graph. In contrast to other approaches, which only indirectly measure reliability, the number of

J. Gottlieb and G.R. Raidl (Eds.): EvoCOP 2004, LNCS 3004, pp. 177–187, 2004.

spanning trees in a graph is a more accurate measurement for the all-terminal reliability of a network [8]. We present empirical results that show that the proposed heuristic outperforms a standard greedy repair heuristic by finding better solutions and using a lower computational effort.

In the following section we give a short problem description. Section 3 investigates different approaches to consider reliability constraints in GA design. In section 4 we discuss the deficites of existing approaches and propose an approach based on the number of spanning trees. Experimental results and a comparison to a simple greedy heuristic are presented in section 5.2. The paper ends with concluding remarks.

2 Problem Definition

For the reliable communication network design problem (RCND) [2], a network topology with minimal cost must be found that satisfies a given reliability constraint. This problem has been proven as NP-hard [3], and several GA-approaches have been proposed for this problem. [9] introduced a branch and bound algorithm minimizing network costs under a reliability constraint. Later, Dengiz et al. proposed a GA [5] using a penalty function to incorporate the reliability constraint directly into the fitness function, as well as a simulated annealing approach [10]. [11] extended this work and developed a parallel GA for larger problem instances. [12] presented a GA with multiple reliability constraints. [6] did not incorporate the reliablity contraint into the fitness function but used a problem specific representation and adapted GA operators.

The RCND problem can be defined as follows: an undirected graph is denoted as $G = (V, E)$, $n = |V|$ denotes the number of nodes, and $m = |E|$ denotes the number of edges of G. It is assumed that the location of each node is given a priori and all nodes are perfect reliable. The degree $d(i)$ of a node i is the number of edges that are connected to node i. For each possible edge $e_{ij} \in E$ the corresponding costs c_{ij} and reliability r_{ij} are known. The probability $1 - r_{ij}$ that the edge e_{ij} fails is statistically independent. A graph G is n-connected if there are at least n edge-disjoint paths between all pairs of nodes i and j. The objective function of the problem is:

$$C(G) = \sum_{i=1}^{n} \sum_{j=i+1}^{n-1} c_{ij} x_{ij} \rightarrow min \qquad \text{subject to: } R(G) \geq R_0, \tag{1}$$

where $C(G)$ is the total cost of the network G and c_{ij} is the costs for an edge connecting node i and j. The variable $x_{ij} \in (0, 1)$ indicates whether edge $e_{ij} \in E$. $R(G)$ is the all-terminal reliability that is the probability that the network G is still connected (even if some of the edges $e_{ij} \in E$ fail). The calculation of the all-terminal reliability has been proven as NP-hard [4]. Exact algorithms for calculating the all-terminal realiability for networks with a low number of nodes

have been proposed by [1,13]. For larger networks, monte carlo-based estimations of the all-terminal reliability [14] are more appropriate. It was shown in [8] that the number of spanning trees in G is an appropriate measurement for the all-terminal reliability (a network G is still connected (and reliable) as long as there is at least one spanning tree in G).

3 Considering Reliability Constraints in Genetic Algorithms

Standard GAs are not able to handle additional problem constraints. Therefore, much research has been focused on how to consider constraints in GA design. It can be distinguished between two different approaches on how to deal with constraints [15,16]. Firstly, indirect constraint handling techniques consider constraints by modifications of the fitness functions. Violations of constraints lead to a lower fitness value (penalty) of the candidate solution. Secondly, direct constraint handling techniques modify the structure of the GA. In principle, there are four different approaches:

- Leave invalid solutions in the population.
- Eliminate infeasible solutions from the population.
- Prevent infeasible solutions by problem-specific representations and operators.
- Repair infeasible candidate solutions.

There are some approaches that have no explicite mechanisms to consider additional constraints but to some extent accept invalid solutions [17]. They hope that the best solution at the end of the run is valid. However, such approaches can only be used if the number of invalid solutions is low.

Other GA approaches eliminate invalid solutions that are generated during a GA run. This approach is only possible if the number of invalid solutions is low. Furthermore, there is the problem that the removal of infeasible candidates solutions may take valuable genetic material from the population that might produce high-quality offspring after recombination and mutation [16,18].

After discussing in general the first two simple direct constraint handling techniques, we focus in the next subsection on the remaining direct, as well as indirect, approaches in the context of the RCND problem.

3.1 Penalty Functions

The indirect constraint handling by penalties as suggested in [19] incorporates a constraint into the fitness function. This transforms a constraint optimization problem to an unconstrained problem by adding penalties for constraint violations to the fitness value of a solution. When using penalties, infeasible solutions remain in the population and their genetic material can be used. Using penalties requires a well-designed penalty function that does not generate new local optima, or let global optima become suboptimal [20].

In the context of the RCND problem, [5] proposed a fitness function with a quadratic penalty term. The objective function from equation 1 becomes:

$$C'(G) = \sum_{i=1}^{n} \sum_{j=i+1}^{n-1} c_{ij}x_{ij} + \delta * (c_{max}(R(G) - R_0))^2$$
$$\delta = \begin{cases} 0, & \text{if } R(G) \geq R_0 \\ 1, & \text{if } R(G) < R_0 \end{cases} \quad (2)$$
$$c_{max} = \max_{e_{ij} \in E}(c_{ij})$$

This problem formulation uses a quadratic penalty term. Additionally, [5] used a repair heuristic which ensures that the degree of all nodes is larger than one ($d(i) \geq 2, \forall i \in V$).

Table 1. Results of GAs using the penalty approach from equation 2

test instance	r_{ij}	R_0	$C(G_{opt})$	$C'(G'_{opt})$	$R(G'_{opt})$
8 nodes - network 1	0.9	0.90	208	194.222	0.899198
	0.9	0.95	247	223.073	0.949009
8 nodes - network 3	0.9	0.9	211	211	0.902212
	0.9	0.95	245	233.067	0.949948

To check if this penalty approach results in correct solutions, we implemented a GA with the fitness function from equation 2. For the experiments we used a steady state GA with a binary representation, uniform crossover without mutation, and an exact reliability evaluation based on [1].

Table 1 shows the results of our experiments using the proposed penalty function for selected 8 nodes test problems using $r_{ij} = 0.9$ and $R_0 = 0.9$ resp. $R_0 = 0.95$. The two test instances (network 1 and network 3) are taken from [5]. We show the total cost of the correct optimal solution $C(G_{opt})$ published in [5], where $R(G_{opt}) \geq R_0$, the lowest found cost $C'(G'_{opt})$ of the network G'_{opt} according to equation 2, and the corresponding all-terminal realiability $R(G'_{opt})$. It can be seen that using equation 2 can result in solutions G'_{opt} that have lower cost $C'(G'_{opt}) < C'(G_{opt})$, but violate the reliability constraint ($R(G'_{opt}) < R_0$). Only for one instance (network 3, $r_{ij} = 0.9$, and $R_0 = 0.9$), could a valid solution be found. For the other problems the penalty from equation 2 is too low to ensure a valid solution. As a result the total fitness for an infeasible network G'_{opt} can be smaller than the fitness for the cheapest feasible network G_{opt}. These examples show that an unfavorable design of penalty functions may cause the solutions with lowest fitness to be infeasible. Summarizing the results, the proposed penalty function from [5] does not work in an effective way and can result in invalid solutions.

3.2 Problem-Specific Representations and Operators

Most standard GAs use binary representations and standard operators like n-point or uniform crossover. When applying such standard operators to valid solutions encoded with a standard representation, the resulting offspring can be invalid. This situation can be avoided by using either problem-specific representations, or operators that consider the constraint at hand.

We want to give two examples for network problems where the optimal solution should be a tree. Trees are a special variant of fully connected graphs G where $|E| = |V| - 1$. The use of the problem-specific Prüfer number representation [21] allows us to consider the constraint that valid solutions are trees. Another possibility to consider this constraint is using direct representations and problem-specific operators (e.g. [22]). The problem-specific operators ensure that only valid solutions (trees) can be created.

[6] presented problem-specific crossover and mutation operators for the RCND problem. The crossover operator randomly exchanges one link between two parents. If the offspring does not satisfy the reliability constraint, an additional heuristic is applied such that the order of each node is greater than one ($d(i) \geq 2, \forall i \in V$). The mutation operator searches for two rings in the graph that share only one common node. To reduce the cost $C(G)$ of the network, it merges the two rings to one single ring. If the parent is a network, where $d(i) \geq 2\ \forall i \in V$, the offspring is also a network with $d(i) \geq 2$.

4 Repair Heuristics

This section shows some deficits of exisiting repair heuristics for the RCND problem, and proposes a new heuristic based on counting spanning trees.

4.1 Deficits of Existing Approaches

When using standard GA operators, problem-specific heuristics can be used to repair invalid solutions violating constraints. A repair heuristic changes candidate solutions such that they become feasible [16]. Two different repair strategies can be distinguished: The Lamarkian approach replaces the parental individual by the offspring. The Baldwinian approach leaves the individual untouched but only its fitness is replaced by the fitness of the repaired solution.

[6] and [5] introduced greedy repair heuristics for the RCND problem. An individual is repaired such that all nodes have at least the degree two ($d(i) \geq 2, \forall i \in V$). The repair strategies used the degree of the nodes as a measurement of the all-terminal reliability of the network. However, a network with $d(i) \geq 2, \forall i \in V$, is not always 2-connected as can be seen for example in Figure 1. If the edge $e_{3,4}$ fails, the network is separated into two unconnected network components. Although all

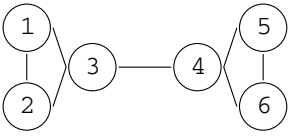


Fig. 1. Example tree with $d(i) \geq 2, \forall i \in V$

nodes in the original network have degree larger than one, it is already disconnected if only one link fails. This example illustrates that the repair procedures proposed in [5,6] only use a weak reliability measure. To design networks based on the all-terminal reliability, more accurate measurements of reliability are necessary.

4.2 Spanning Tree Counting Repair Heuristic

In the previous paragraphs we have discussed the deficits of existing approaches solving the RCND problem. Therefore, inspired by the reliability improvement procedure proposed by [8], we introduce a GA using a spanning tree counting (STC) repair heuristic. As the exact calculation of the all-terminal reliability causes high computation effort, [8] use the number of spanning trees in the graph G as a measurement of all-terminal reliability. It was shown that the number of spanning trees in the graph is a good approximation for the all-terminal reliability.

The basic idea of the STC repair heuristic is to add these edges to the graph that maximize the reliability (number of spanning trees in the network) with minimal additional costs. Consequently, the STC repair heuristic calculates for the cheapest edges that are not in the network G, the possible increase of spanning trees if these edges are added:

1. Sort all links $e_{ij} \notin E$ according to the corresponding edge costs c_{ij}. $i = 0$.
2. Insert the i-cheapest edge e_{ij} temporarily into G and calculate the ratio $s_{ij} = c_{ij}$/increase in number of spanning trees in G. $i = i + 1$.
3. If $i < t$ continue with step 2.
4. Add edge $e_{ij} \notin E$ with highest corresponding s_{ij} to G.
5. Calculate $R(G)$.
6. If $R(G) < R_0$, then continue with step 4.

In step two and three the heuristic calculates for the cheapest edges, the ratio between the cost of the edge, and the increase in the number of spanning trees. The increase in the number of spanning trees can be calculated with low computational effort by a simple update procedure [8]. The number of edges that are investigated is limited by t, where $t < n(n-1)/2 - |E|$. In step four to six edges with highest improve ratio s_{ij} are iteratively added to G until the reliability constraint is fullfilled.

5 Experiments

5.1 Experimental Design

For our experiments we use a steady state GA with a binary representation of length $l = n(n-1)/2$. The existence of an edge in G is encoded by 1, its absence by 0. The GA uses one-point crossover and bit-flipping mutation. The initial population consists of randomly created 2-connected graphs. The initialization

routine firstly creates a random spanning tree and then randomly adds links until the graph is 2-connected.

As the effort for calculating network reliability is high, we used several techniques to speed up reliability evaluation. As a first step in calculating the all-terminal reliability of a network we determined an upper bound $R_{up}(G)$ for the reliability of a network G using a method proposed by [23]. If $R_{up}(G) < R_0$ the network can not fullfill the reliability constraint and it is not necessary to calculate the reliability exactly. Only if $R_{up}(G) \geq R_0$, we calculate $R(G)$ exactly using a method proposed by [1]. When using this method, we already get a measurement of the all-terminal realiability during the run. We stop the exact calculation as soon as the reliability constraint R_0 is satisfied. Finally, to avoid calculating the reliability of networks G that have been evaluated previously, we store the reliability of all graphs using a hash table. For all new individuals the hash is searched if the network reliability has already been calculated.

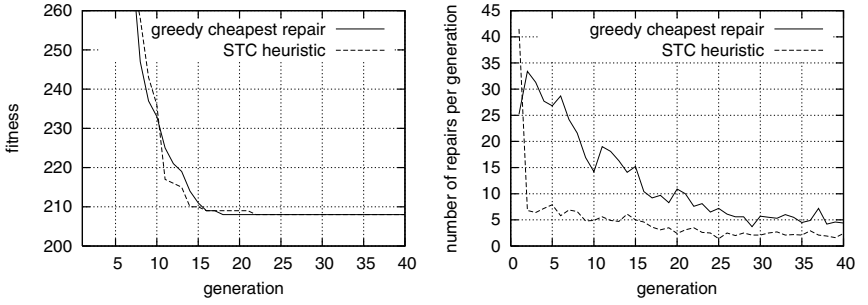
If the network reliability of a network is too low ($R(G) < R_0$) the STC heuristic repairs the network and adjusts the chromosome according to the new network. For comparison we have implemented an additional greedy cheapest repair procedure. This heuristic adds the cheapest edge to the network until the graph satisfies the constraint. Unlike the STC heuristic, it does not consider reliability (increase in number of spanning trees) when choosing edges. After constructing a valid solution by a repair heuristic the fitness of the individual is calculated according to equation (1). This approach ensures that we have only valid solutions.

In our experiments we use a steady state GA with 50% replacement, a crossover probability $p_{cross} = 0.9$, a mutation probability $p_{mut} = 0.01$, a population size of 100, an edge reliability $r_{ij} = 0.9, \forall e_{ij} \in E$, and a reliability constraints $R_0 = 0.9$ and $R_0 = 0.95$. The GA stops after 250 generations or convergence. For each test instance we performed ten independent runs.

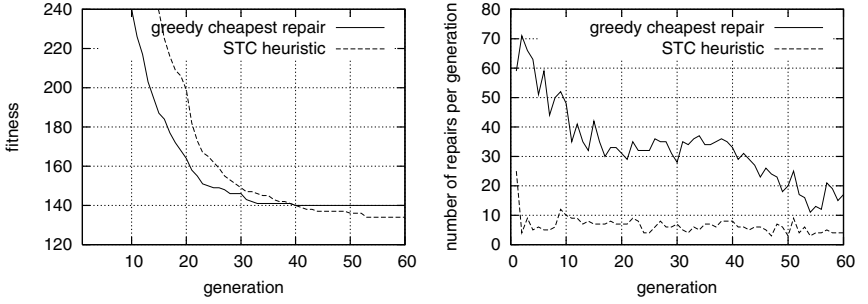
5.2 Results

Both heuristics have been tested with network problems (8, 10 and 11 nodes) taken from [5], and a new test problem for the 15 largest German cities. We compare the quality of the solutions and the number of repair operations that are necessary for finding high quality solutions. As after each repair operation a reliability check has to be performed, and the reliability checks are computationally demanding, the number of repair operations impacts the running time of the GA. Unfortunately it is not possible to compare our approach directly to the results from [5], because they penalize invalid solutions using the objective function from equation 2, and repair invalid solutions with regard to the degree of the nodes ($d(i) \geq 2, \forall i \in N$). The penalty approach can not be used for a comparison as it can result in invalid solutions (compare section 3.1).

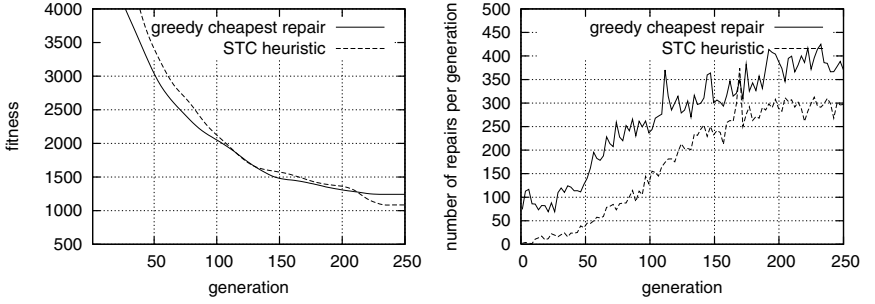
Figure 2 compares the results for the STC heuristic and the greedy cheapest repair heuristic. The plots show the fitness of the best solution and the number of repair operations over the number of generations for the 8 nodes (Figure 2(a)), 10 nodes (Figure 2(b)), and the new 15 nodes (Figure 2(c)) problem. All values



(a) 8 node



(b) 10 node



(c) 15 nodes

Fig. 2. Fitness of the best solution (left) and number of repair operations (right) over the number of generations. The plots show that the STC heuristic finds better solutions and needs less repair operations

are averaged over ten runs. The plots show that the STC heuristic converges more slowly towards high-quality solutions, but always finds better solutions at the end of the run. The plots for the number of repairs show that the STC repair heuristic needs significantly less repairs. Therefore, the STC heuristic is for the 15 nodes problem much faster in comparison to the greedy cheapest repair heuristic (compare also Table 2). This advantage in running time can not be observed for the 8 nodes and 10 nodes problem. The used all-terminal reliability calculation needs for both test problem a low computational effort, therefore the additional repairs have only little impact on the GA running time.

Table 2. Comparison of STC and greedy cheapest repair heuristic

nodes	test problem	R_0	optimum	method	best found	repairs	t_{conv}
8	1	0.9	208	STC	208	158	4 sec
				greedy	208	300	4 sec
8	1	0.9	203	STC	203	658	4 sec
				greedy	203	1548	4 sec
8	2	0.95	194	STC	194.9	595	4 sec
				greedy	207.5	1749	4 sec
8	3	0.9	211	STC	211.6	526	4 sec
				greedy	211.8	2416	4 sec
8	3	0.95	197	STC	198.7	583	4 sec
				greedy	199.1	2789	4 sec
8	4	0.9	291	STC	294.4	700	4 sec
				greedy	296.5	2071	4 sec
8	4	0.95	276	STC	282.1	618	4 sec
				greedy	283.2	2956	4 sec
10	1	0.9	131	STC	134	428	28 sec
				greedy	140.5	1884	29 sec
10	2	0.9	154	STC	155.8	585	27 sec
				greedy	166.7	2488	27 sec
10	2	0.95	136	STC	137.6	1379	26 sec
				greedy	145.5	2479	28 sec
10	3	0.9	267	STC	268	1339	26 sec
				greedy	269.1	1987	26 sec
10	3	0.95	236	STC	243.2	1650	20 sec
				greedy	244.7	2891	20 sec
11		0.9	246	STC	246.7	747	49 sec
				greedy	248.4	1691	50 sec
15		0.9	1006.9	STC	1086.6	44025	9120 sec
				greedy	1217.24	68358	44230 sec

Table 2 summarizes the results for the two heuristics and shows the optimal solution, the average best solution found at the end of a GA run, the average number of repair operations that are necessary to find the best solution, and the average running time t_{conv}. The optimal solutions for the 8, 10 and 11 nodes problem have been published in [5]. The optimal solution for the 15 nodes test

problem is the best ever found solution from a GA using the STC heuristic. As before a GA using the STC heuristic finds better solutions with lower computational effort. The additional effort of the STC heuristic for calculating the potential increase in the number of spanning trees in G (compare section 4.2) is low, as it can be computed as the determinant of the node degree matrix [24].

6 Conclusions

This paper investigates existing GA approaches for the reliable communication network design (RCND) problem and proposes a heuristic repair approach based on the number of spanning trees in a network. The analysis of existing approaches for solving the RCND problem reveals some deficits. The penalty approach from [5] can result in invalid solutions and the greedy repair heuristics introduced by [6] and [5] repair invalid solutions according to the degree of the nodes and do not consider the all-terminal reliability of the network.
Therefore, we present a spanning tree counting (STC) repair heuristic that can be combined with standard GAs. This heuristic considers the number of spanning trees in a graph as a more meaningful reliability measure when repairing invalid solutions. The empirical results show that the STC heuristic outperforms a greedy cheapest repair heuristic that considers only the cost of links. The STC heuristic allows only valid solutions and finds in comparison to the greedy cheapest repair heuristic better solution using less computational effort.

References

1. Yubin Chen, Jiandong Li, and Jiamo Chen. A new algorithm for network probabilistic connectivity. In *Proceedings of the IEEE military communication conference*, Piscataway, NJ, 1999. IEEE Service Center.
2. Robert R. Boorstyn and Howard Frank. Large-scale network topological optimization. *IEEE Transactions on Reliability*, 25:29–37, 1977.
3. M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Fransisco, 1979.
4. Li Ying. Analysis method of survivability of probabilistic networks. *Military Communication Technology Magazine*, 48, 1993.
5. B. Dengiz, F. Altiparmak, and A. E. Smith. Local search genetic algorithm for optimal design of reliable networks. *IEEE Trans. on Evolutionary Computation*, 1(3):179–188, 1997.
6. Sheng-Tzong Cheng. Topological optimization of a reliable communication network. *IEEE Transactions on Reliability*, 47(3):225–233, 1998.
7. Baoding Liu and K. Iwamura. Topological optimization model for communication network with multiple reliability goals. *Computer and Mathematics with Applications*, 39:59–69, 2000.
8. N. Fard and Tae-Han Lee. Spanning tree approach in all-terminal network reliability expansion. *computer communications*, 24:1348–1353, 2001.
9. Rong-Hong Jan, Fung-Jen Hwang, and Sheng-Tzong Chen. Topological optimization of a communication network subject to a reliability constraint. *IEEE Transactions on Reliability*, 42(1):63–70, 93.

10. B. Dengiz and C. Alabas. A simulated annealing algorithm for design of computer communication networks. In *Proceedings of World Multiconference on Systemics, Cybernetics and Informatics, SCI 2001*, volume 5, 2001.
11. Benjamin Baran and Fabian Laufer. Topological optimization of reliable networks using a-teams. In *Proceedings of World Multiconference on Systemics, Cybernetics and Informatics - SCI '99 and ISAS '99*, volume 5, 1999.
12. B.Liu and K. Iwamura. Topological optimization models for communication network with multiple reliability goals. *Computers and Mathematics with Applications*, 39:59–69, 2000.
13. K.K. Aggarwal and Suresg Rai. Reliability evaluation in computer-communication networks. *IEEE Transactions on Reliability*, 30(1):32–35, 1981.
14. E. Manzi E., M. Labbe, G. Latouche, and F. Maffioli. Fishman's sampling plan for computing network reliability. *IEEE Transactions on Reliability*, 50(1):41–46, 2001.
15. Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, 3 edition, 1996.
16. B.G.W. Craenen, A.E. Eiben, and E.Marchiori. How to handle constraints with evolutionary algortihms. In Lance Chambers, editor, *Practical Handbook Of Genetic Algorithms: Applications*, pages 341–361. Chapman & Hall/CRC, 2000.
17. L. Davis, D. Orvosh, A. Cox, and Y. Qiu. A genetic algorithm for survivable network design. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 408–415, San Mateo, CA, 1993. Morgan Kaufmann.
18. Steven Orla Kimbrough, Ming Lu, David Harlan Wood, and D.J. Wu. Exploring a two-population genetic algorithm. In Erick Cantu-Paz et al, editor, *Proceedings of the Genetic and Evolutionary Computation Conference 2003*, pages 1148–1159, Berlin, 2003. Springer-Verlag.
19. D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, MA, 1989.
20. Jens Gottlieb. *Evolutionary Algorithms for Constrained Optimization Problems*. PhD thesis, Technische Universität Clausthal, Institut für Informatik, Clausthal, Germany, 1999.
21. H. Prüfer. Neuer Beweis eines Satzes über Permutationen. *Archiv für Mathematik und Physik*, 27:742–744, 1918.
22. Günther R. Raidl and Bryant A. Julstrom. Edge-sets: An effective evolutionary coding of spanning trees. *IEEE Transactions on Evolutionary Computation*, 7(3):225–239, 2003.
23. A. Konak and A. Smith. An improved general upperbound for all-terminal network reliability. Technical report, University of Pittsburgh, 1998.
24. C.J. Colbourn. *The Combinatorics of Network Reliability*. Oxford University Press, 1987.