SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 8, NO. 4
ISSN: 2054 -7420

A Genetic Algorithm to Solve the Robust Design Problem for A Flow Network With Node Failure

Noha Hamdy Radwan, Moatamad Refaat Hassan, Mohamed Eid Hussein
Computer Science and Mathematics Department, Faculty of science, Aswan University, Aswan, Egypt;
nohahamdy9018@gmail.com; mr.hassan@sci.aswu.edu.eg; dr_meid@sci.aswu.edu.eg

ABSTRACT

The robust design problem in a flow network is defined as search optimal node capacity that can be assigned such that the network still survived even under the node's failure. This problem is considered as an NP-hard. So, this paper proposes a genetic algorithm-based approach to solve it for a flow network with node failure. The proposed based genetic approach is used to assign the optimal capacity for each node to minimize the total capacities and maximize the network reliability. The proposed approach takes the capacity for each critical node should have the maximum capacity (usually equals to the demand value) to alleviate that the reliability to drop to zero. Three network examples are used to show the efficiency of our algorithm. Also, the results obtained by our approach are compared with those obtained by the previous approximate algorithm.

Keywords: Flow network, robust design, node failure, genetic algorithm.

1 Introduction

Reliability evaluation is an important index to reorder the scheduling in many network systems such as the communication system [1]. The reliability of a computer network is defined as the probability that any source node can successfully communicate with any terminal node when no flow happens [2-3]. In the case of existing a flow between nodes, the system reliability of a flow network is defined as the probability that the maximum flow path throw (nodes or arcs) is not less than the given demand (d) [4-6]. Lin in [6] presented a simple algorithm to calculate system reliability based on generating all lower boundary points for the given demand d. Lin in [7] constructed a stochastic-flow network to model the information system. Lin in [8] proposed a new performance index, the probability that the upper bound of the system capacity equals the demand vector subject to the budget constraint, to evaluate the quality level of a multicommodity limited-flow network. Statitsatin and Kapur [9] presented an algorithm to search for lower boundary points and used it for computing the exact reliability.

In [10] the research discussed the capacity assignment problem and gave the definition to the critical node. Proved that, if the critical node failed then the system reliability drop to zero. In general, the capacity assignment problem (CAP) is defined as the search for the optimal capacities that minimize the delay in the network and maximize its reliability [11]. Structural analysis is used to identify the nodes' structural and locate critical nodes to avoid the system reliability don`t drop to zero [12]. The robust design is defined as an issue in quality engineering where the designed product function well even in a versatile

DOI: 10.14738/tnc.84.8949
Publication Date: 17th September 2020
URL: http://dx.doi.org/10.14738/tnc.84.8949

environment. However, in the context of a stochastic-flow network, it means the network function well even in a node`s failure situation [12]. The robustness of an assignment is therefore recognized as the fact the critical nodes generated after the assignment equal to the structural important nodes [12-13]. The robust design problem (RDP) in a capacitated flow network is to search for the minimum capacity assigned to each edge such that the network still survived even under the edge`s failure [14].

A genetic algorithm (GA) is a heuristic search method used in optimization problems. [4] present a GA to compute the reliability of a stochastic-flow network in which each arc or node has several capacities and may fail. To solve reliability optimization problems, many based GA approaches were proposed such as [4, 14-16]. Also, GA is used to solve multiple objective optimization problems [17-21]. In addition, GA is used to solve the capacity assignment problem and the objective is to minimize the total capacities while meeting the network reliability requirement [22].

In this paper, we propose an approach based on GA to solve the robust design problem, i.e. to search the maximum reliability such that the sum of the assigned capacities is minimized.

The paper is organized as follows: We illustrate notations and assumptions in section 2. Section 3 presents the problem description. Section 4 explains the problem formulation. Section 5 describes the GA components. The proposed GA algorithm given in section 6. Studied cases shown in section 7. Finally, we present the conclusion in section 8.

2 Notations and assumptions

np	No. of paths
$\mathrm{mp_j}$	Minimal paths no. j, $\mathrm{j} = 1, 2, \dots, \mathrm{np}$
d	The demand.
neq	No. of components (nodes).
n	The node.
F	Flow vector; $\mathrm{F} = (\mathrm{f_1}, \mathrm{f_2}, \dots, \mathrm{f_{np}})$.
X	Capacity vector; $\mathrm{X} = (\mathrm{x_1}, \mathrm{x_2}, \dots, \mathrm{x_{neq}})$.
r	The probability.
$\mathrm{R_d}$	System reliability to the given demand d.
NP	Number of chromosomes.
NG	Number of genes equals to neq.
maxgene	The maximum number of generations.
cr	The GA crossover rate.
mr	The GA mutation rate.

3 Problem Description

3.1 The reliability evaluation

Given the demand d, the system reliability R_d is defined by, Lin [6]

$$R_d = Pr\{X \mid V\ (X)\ \geq\ d\} \tag{1}$$

Where X is a lower boundary point for d. The RSDP procedure presented by Zue et al. (2007), [1] is used to evaluate R_d according to eq. 1.

And X can be deduced from $F = (f_1,\ f_2,\ f_3, \dots, f_{np})$ by using the following equation:

$$x_i = \sum_{j=1}^{np}\{f_j \mid n_i \in\ mp_j\} \text{ for each } i = 1, 2, \dots, neq \tag{2}$$

when F generate by satisfies the flowing two constraints:

$$\sum_{j=1}^{np} f_j \mid n_i \in mp_j\} \leq M^i \text{ for } i = 1, 2, \dots, neq \quad (3)$$

$$\sum_{j=1}^{np} f_j = d \quad (4)$$

3.2 The capacity assignment

The analysis of a network structure is important to assignment the capacity. At first, we need to define the concept node covering to analyses the network structural, which extends to the definition of structural impact factor (SIF) S_i for node n_i.

3.3 Coverage node

When $n_i, n_j \in N$. and $n_j \subseteq n_i$, n_j is said to be covered by n_i if no flow pass through $n_j$when n_i failed, [12-14].

3.4 Critical node

The node n_i called critical nodes if and only if $R_{d,i}$ is zero [12-14].

3.5 Probabilities states calculation for each node

At first, we should be defining the probability for $\Pr\{x_i\}$ of node $n_i$to calculate R_d. we assuming find $c_i$components for node n_i to produce the corresponding capacities. Each component has the reliability of r_i, [12]. Then the probability for the current capacity x_i is $Pr\{x_i\}$ and is denoted as:

$$Pr\{x_i = k\} = \binom{c_i}{k} r_i^k (1 - r_i)^{c_i - k} \quad (5)$$

4 Problem formulation

Let $M = (M^1, M^2, \dots, M^{neq})$ as assigned capacities to the set of nodes $(n_1, n_2, \dots, n_{neq})$. The mathematical formulation of the problem is:

$$\text{Minimize } \mathbb{C} \quad (6)$$

S.t.

$$R_d \text{ is maximized} \quad (7)$$

Where $\mathbb{C} = \sum_i^n M^i$ and R_d is reliability corresponding to the assigned capacities under demand d. M^iranges from 1 to d, except for critical nodes [Lin (2008)], $M^i = d$.

5 The GA components

The following subsections depict different components of the proposed GA.

5.1 Representation

A chromosome representation is indispensable to characterize each chromosome in the GA algorithm. The chromosome M is symbolized by a series of length (neq), where (neq) is the number of nodes as shown in Figure 1.

$$(M^1, M^2, \dots, M^{neq})$$

Figure 1. Chromosomal representation

Copyright © Society for Science and Education, United Kingdom

5.2 Crossover

Is a genetic operator used to collect the genetic information of two parents called parent1 and parent2 to generate new offspring called offspring1 and offspring2.

In this research We will use Single Point Crossover, a crossover point is randomly generated which determines the point for swap of information between two parents to generate two offspring as shown in Figure 2.

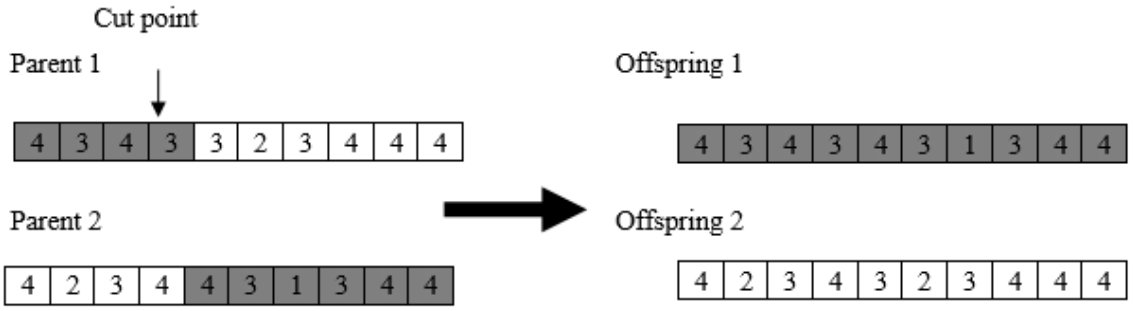


Figure 2. Crossover process

5.3 Mutation

Mutation takes place after crossover is done. It makes the changes randomly to one "genes" to creates a new offspring. In this research, we will use Creep Mutation as shown in Figure 3. In creep mutation, the gene is selected random and the value of it is changed with a random value between $[1: d]$ unless the value of it.

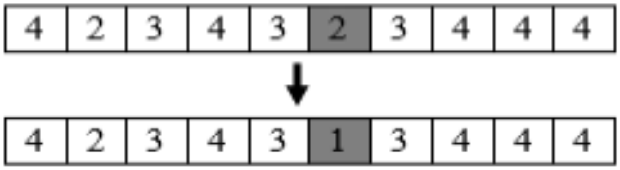


Figure 3. Mutation process

5.4 Fitness function

We will use the total sum of capacities $\mathbb{C}$ for each chromosome i as a fitness function, i.e. $fit(i) = \mathbb{C}$. For the selection process, the fitness function is normalized as follows:

Calculate the sum of fitness for all chromosomes in the population, sum_fit.

For i =1 to NP

$$normalize_fit(i) = fit(i)/sum_fit$$

End for

5.5 Selection

Selection process takes the fittest solutions by giving it a high percentage when choosing the next generation and thus leads to choosing the best solution to the problem. In this research, we will use Roulette wheel selection to choose two parents based on cumulative sum ($cumsum$) as follows:

Begin

For i =1 to NP

For j = i to NP

cumsum(i) = cumsum(i) + normalize_fit(j)

End for j

End for i

Generate a random number μϵ[0, 1]

For i = 1 to NP

If μ >cumsum(i)

Parent1=i-1.

End if

End for i

Generate a random number μϵ[0, 1]

For i=1 to NP

If μ >cumsum(i)

Parent2=i-1.

End if

End for i

End

6 The proposed GA algorithm

Start

Input: $d, neq, MPs, r_i, and\ components'\ informations.$

Determine the critical nodes in the network.

Set the GA parameters: $NP, NG,\ maxgene, cr, and\ mr.$

Generate the initial population randomly.

Evaluate the initial population.

While $g\ <=\ maxgene$, do

While $p\ <=\ NP$, do

Copyright © Society for Science and Education, United Kingdom

Select two chromosomes.

Generate new offspring after applying crossover and mutation.

$p = p + 1$

end do

Evaluate the current population.

Save the best solutions.

$g = g + 1$

End do

Report the best solutions found.

End

7 Studied Cases

In the following subsections, the proposed GA has been applied to three networks. We studied and observed the results in the case of varying each GA parameter. We found that the best values for these parameters are: $Maxgene = 30, NP = 10, cr = 0.95, and\ mr = 0.05$.

7.1 Eight-node network

This network has eight nodes as shown in Figure 4. The network has eight minimal paths as follows:

mp_1 = $\{n_1, n_2, n_4, n_8\}$, mp_2 = $\{n_1, n_2, n_5, n_8\}$, mp_3 = $\{n_1, n_2, n_6, n_8\}$, mp_4 = $\{n_1, n_2, n_7, n_8\}$, mp_5 = $\{n_1, n_3, n_4, n_8\}$, mp_6 = $\{n_1, n_3, n_5, n_8\}$, mp_7 = $\{n_1, n_3, n_6, n_8\}$, mp_8 = $\{n_1, n_3, n_7, n_8\}$. The nodes' probabilities, r_i = (0.90, 0.88, 0.93, 0.91, 0.89, 0.90, 0.87, 0.92). Table 1 shows the best candidate M with the corresponding $\mathbb{C}$, R_4, execution time.

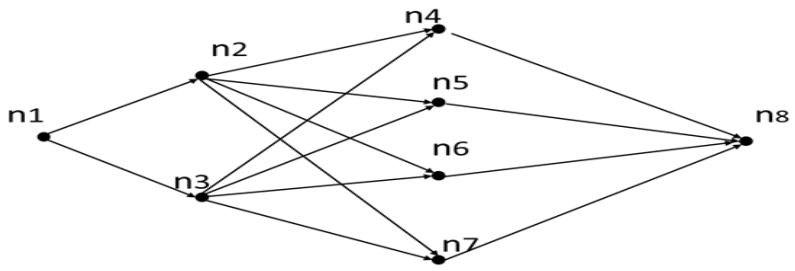


Figure 4. Eight-node network.

Table 1. The results for eight-node network with d=4.

Run no.	The best M	$\mathbb{C}$	R_4	Execution time (in seconds)
1	[4,4,4,3,4,3,1,4]	27	0.469878	0.06
2	[4,4,1,4,4,2,3,4]	26	0.424858	0.05
3	[4,3,4,4,4,3,3,4]	29	0.469165	0.05
4	[4,3,4,3,3,3,4,4]	28	0.469166	0.05
5	[4,3,4,4,4,3,2,4]	28	0.469165	0.05

6	[4,4,4,4,3,3,1,4]	27	0.469880	0.05
7	[4,2,4,4,2,2,1,4]	23*	0.465164	0.05
8	[4,2,4,4,2,4,4,4]	28	0.465196	0.05
9	[4,3,3,2,3,2,4,4]	25	0.463771	0.05
10	[4,4,2,4,1,4,1,4]	24	0.462064	0.05

*The minimum $\mathbb{C}$ found.

7.2 Ten-node network

This network has ten nodes as shown in Figure 5, it has seven minimal paths as follows [12]:

mp_1=$\{n_1, n_2, n_4, n_8, n_{10}\}$,$mp_2$= $\{n_1, n_2, n_5, n_8, n_{10}\}$, mp_3= $\{n_1, n_2, n_6, n_8, n_{10}\}$,

mp_4= $\{n_1, n_2, n_6, n_9, n_{10}\}$,$mp_5$= $\{n_1, n_3, n_6, n_8, n_{10}\}$, mp_6= $\{n_1, n_3, n_6, n_9, n_{10}\}$,

mp_7= $\{n_1, n_3, n_7, n_9, n_{10}\}$. r_i = (0.99, 0.98, 0.97, 0.98, 0.98, 0.99, 0.97, 0.98, 0.97, 0.99). Table 2 shows the best candidate M with the corresponding$\mathbb{C}$, R_4, execution time.

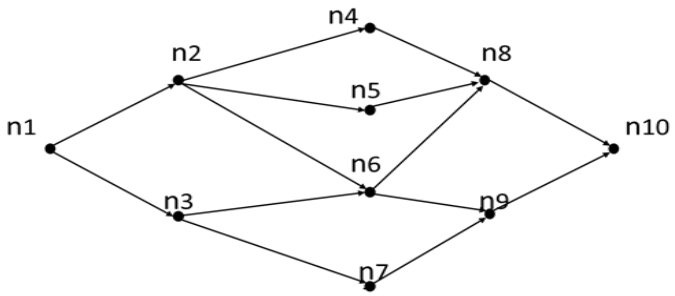


Figure 5. Ten-node network

Table 2. The results for ten-node network with d=4.

Run no.	The best M	$\mathbb{C}$	R_4	Execution time (in seconds)
1	[4,4,3,3,1,1,4,4,2,4]	30*	0.922506	0.05
2	[4,3,4,1,4,2,2,3,3,4]	30	0.922458	0.05
3	[4,2,4,3,3,2,2,3,3,4]	30	0.922027	0.05
4	[4,4,4,4,2,2,1,3,3,4]	31	0.922386	0.05
5	[4,4,2,2,3,4,3,3,2,4]	31	0.917511	0.05
6	[4,4,4,2,1,3,4,4,3,4]	33	0.922734	0.04
7	[4,3,4,1,4,4,1,4,4,4]	33	0.922731	0.05
8	[4,3,4,1,3,4,2,4,2,4]	31	0.922515	0.05
9	[4,4,4,3,1,2,3,3,3,4]	31	0.922478	0.05
10	[4,4,3,3,2,3,1,4,3,4]	31	0.922723	0.06

*The minimum $\mathbb{C}$ found.

The results of the ten-node network are compared with the algorithm presented in [10] as shown in Table 3. We found that the proposed GA succeeds in obtaining minimum $\mathbb{C}$ with maximum R_4.

Table 3. Comparison results with Chen and Lin [10]

Algorithm used	M	$\mathbb{C}$	R_4
Chen and Lin [10]	[4 4 4 2 2 3 3 4 4 4]	34	0.922643
Proposed GA	[4, 4, 1, 3, 1, 4, 1, 3, 3, 4]	28	0.922734

Copyright © Society for Science and Education, United Kingdom

7.3 Thirteen-node network

The network has 30 nodes as shown in Figure 6 and 6 minimal paths as follows:

mp_1= $\{n_1, n_2, n_4, n_6, n_8, n_{10}, n_{12}, n_{14}, n_{16}, n_{18}, n_{20}, n_{22}, n_{24}\}$,

mp_2= $\{n_1, n_3, n_5, n_7, n_9, n_{11}, n_{13}, n_{15}, n_{17}, n_{19}, n_{21}, n_{23}, n_{24}\}$,

mp_3= $\{n_1, n_2, n_4, n_6, n_8, n_{10}, n_{25}, n_{27}, n_{28}, n_{29}, n_{14}, n_{16}, n_{18}, n_{20}, n_{22}, n_{24}\}$,

mp_4 = $\{n_1, n_2, n_4, n_6, n_8, n_{10}, n_{25}, n_{27}, n_{28}, n_{30}, n_{15}, n_{17}, n_{19}, n_{21}, n_{23}, n_{24}\}$,

mp_5= $\{n_1, n_3, n_5, n_7, n_9, n_{11}, n_{26}, n_{27}, n_{28}, n_{30}, n_{15}, n_{17}, n_{19}, n_{21}, n_{23}, n_{24}\}$,

mp_6 = $\{n_1, n_3, n_5, n_7, n_9, n_{11}, n_{26}, n_{27}, n_{28}, n_{29}, n_{14}, n_{16}, n_{18}, n_{20}, n_{22}, n_{24}\}$.

r_i = (0.99, 0.88, 0.93, 0.91, 0.91, 0.89, 0.87, 0.91, 0.90, 0.90, 0.93, 0.92, 0.88, 0.88, 0.87, 0.92, 0.92, 0.93, 0.90, 0.90, 0.91, 0.87, 0.87, 0.99, 0.89, 0.93, 0.90, 0.93, 0.87, 0.89). Tables 4 and 5 describe the results for $d = 4$ and $d = 3$ respectively.

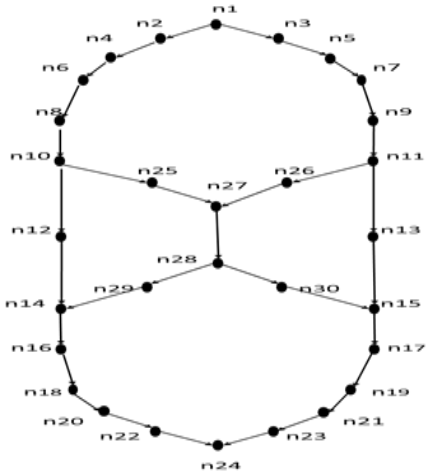


Figure 6. thirty-node network

Table 4. The results for Figure 6 network with $d = 4$

Run no.	M	$\mathbb{C}$	R_4	Execution time (in seconds)
1	[4,1,3,3,3,4,4,4,4,2,4,4,4,3,4,3,4,2,4,3,4,3,4,4,4,2,1,4,4,2]	99	0.361773	0.07
2	[4,2,4,4,3,3,2,3,3,3,3,4,1,2,2,4,3,4,2,3,4,4,4,4,2,4,1,1,3,2]	88	0.149065	0.05
3	[4,2,3,4,4,4,2,2,3,3,3,4,1,3,1,4,2,4,3,4,2,4,3,4,4,2,2,3,4,2]	90	0.183213	0.06
4	[4,4,3,4,4,4,1,4,2,4,2,3,4,4,4,4,1,4,2,3,4,4,3,4,4,1,2,3,2,1]	93	0.317407	0.06
5	[4,2,2,2,3,3,4,2,2,3,3,4,2,3,3,2,4,4,4,4,2,3,2,4,1,4 ,3,4,1,2]	86	0.140364	0.05
6	[4,4,2,3,4,4,2,3,2,4,4,3,2,2,4,4,4,4,3,4,4,4,3,4,1,2 ,4,2,2,1]	93	0.433125	0.05
7	[4,3,1,4,4,4,3,4,1,4,3,4,4,4,2,3,4,3,2,3,3,3,3,4,3,4 ,3,3,3,1]	94	0.248021	0.07
8	[4,3,4,2,3,4,4,2,4,3,3,2,4,4,4,4,4,1,3,3,4,3,3,4,1,2,1,2,3,3]	91	0.248068	0.05
9	[4,2,3,3,3,2,3,2,2,3,3,2,1,4,2,2,3,4,4,3,4,2,3,4,4,4,3,3,2,1]	85*	0.109870	0.06
10	[4,4,4,3,3,4,4,4,3,4,3,1,3,3,3,1,3,4,4,4,4,1,3,4,4,1 ,3,4,3,3]	96	0.211055	0.05

*The best value for $\mathbb{C}$.

Table 5. The results for Figure 6 network with $d = 3$

Run no.	M	$\mathbb{C}$	R_3	Execution time (in seconds)
1	[3,4,3,4,3,3,2,3,2,4,3,1,3,2,3,3,3,3,3,4,2,4,2,3,3,4,3,3,3,4]	90	0.767287	0.05
2	[3,3,4,3,2,4,4,3,1,3,4,3,2,2,3,2,4,1,2,1,3,1,4,3,3,4,2,4,1,1]	80	0.344367	0.05
3	[3,2,3,4,2,3,4,2,3,2,3,2,4,3,3,2,2,2,2,1,4,3,3,3,4,1,2,3,3,3]	81	0.443942	0.05
4	[3,2,1,3,4,2,2,3,4,2,3,4,2,4,3,3,2,2,1,3,3,3,4,3,3,2,3,1,1,3]	79	0.284158	0.05
5	[3,2,2,3,2,4,2,4,4,2,3,1,4,3,2,2,2,3,2,2,2,4,2,3,3,2,4,4,2,1]	79*	0.514976	0.05
6	[3,4,2,2,4,4,4,1,3,4,3,2,3,1,3,1,4,3,4,3,4,1,2,3,3,3,3,1,1,3]	82	0.348693	0.05
7	[3,4,3,3,1,3,4,2,3,4,2,3,4,1,3,2,4,4,3,2,4,2,3,3,1,4,3,4,3,2]	87	0.433159	0.05
8	[3,3,2,2,4,3,1,4,2,3,3,1,4,3,4,1,4,4,2,4,4,3,4,3,1,3,4,4,2,3]	88	0.357166	0.05
9	[3,2,1,2,4,3,2,4,3,4,2,4,3,3,4,3,2,2,4,3,2,2,4,3,4,3,1,1,2,4]	84	0.424574	0.05
10	[3,3,4,2,2,4,2,2,2,3,3,4,3,1,4,3,2,2,2,4,3,1,3,3,4,3,3,4,4,2]	85	0.355394	0.05

*The best value for $\mathbb{C}$.

8 Conclusion

This paper proposed a based GA approach for solving the robust design problem of a flow network with node failure. The proposed approach starts by locating the critical nodes and then starting the searching process to determine the minimum total capacities such that the reliability is maximized. Also, the results obtained for the ten-node network is compared with that obtained by Chen and Lin [10] as shown in Table 3. It is observed that the results of the proposed approach are better than the ones obtained by Chen and Lin [10]. There is no comparison to demonstrate for the eight-node and thirteen-node networks because they are never discussed previously with node failure.

REFERENCES

[1]. Ming J. zuo, Zhigang Tian and Hong-Zhong Huang, An efficient method for reliability evaluation of multistate networks given all minimal path vectors, IIE Transactions, 2007. 39: p. 811-817.

[2]. Aggarwal K. K. and Rai Suresh, Reliability evaluation in computer communication networks, IEEE Transactions on Reliability, 1981. 30(1).

[3]. A. Younes, Spanning trees and reliability of computer networks, Egyptian Informatics Journal, 2005. 6(1).

[4]. A. Younes and M. R. Hassan, A genetic algorithm for reliability evaluation of a stochastic-flow network with node failure, International Journal of Computer Science and Security (IJCSS), 2011. 4(6): p. 528-537.

[5]. Lin J., Jane C. and Yuan J., On reliability evaluation of a capacitated-flow network in terms of minimal path sets, Networks, 1995. 25: p.131-138.

[6]. Y. K. Lin, A simple algorithm for reliability evaluation of a stochastic-flow network with node failure, Computer and Operations Research, 2001. 28: p. 1277-1285.

[7]. Y. K. Lin, Reliability evaluation for an information network with node failure under cost constraint, IEEE Transactions on Systems, Man Cybernetics, part A: System and Humans, 2007. 37(2): p. 180-188.

[8]. Y. K. Lin, System reliability of a limited-flow network in multicommodity case, IEEE Transactions on Reliability, 2007. 56(1): p.17-25.

Copyright © Society for Science and Education, United Kingdom

[9]. Satitsatian Sarintip and Kapur Kailash C., An algorithm for lower reliability bounds of multistate two-terminal networks, IEEE Transactions on Reliability, 2006. 55(2): p. 199-206.

[10]. S. G. Chen and Y. K. Lin, Capacity assignment for a stochastic-flow network based on structural importance, Asian Int. Workshop Adv. Reliab. Model. Taichung, Taiwan, 2008.

[11]. A. F. Zantuti, Algorithms for capacities and flow assignment problem in computer networks, 19th International Conference on Systems Engineering, 2008. p. 315-318.

[12]. Shin-Guang Chen and Yi-Kuei Lin, An Approximate Algorithm for the Robust Design in a Stochastic-Flow Network, Communications in Statistics-Theory and Methods, 2010. 39: p. 2440-2454.

[13]. Y. Niu, X. Xu, C. He, D. Ding and Z. Liu, Capacity Reliability Calculation and Sensitivity Analysis for a Stochastic Transport Network in IEEE Access, 2020. 8: p. 133161-133169.

[14]. Shin-Guang Chen, An optimal capacity assignment for robust design problem in capacitated flow networks Applied Mathematical Modelling ,2012. 36: p. 5272-5282.

[15]. D. L. Deeter and A. E. Smith, Heuristic Optimization of Network Design Considering All – Terminal Reliability in proc. Reliability and Maintainability Symp , 1997. p. 194-199.

[16]. Berna Dengiz Fulya Altiparmak and Alice E. Smith, Local search genetic algorithm for optimal design of reliable network IEEE Transactions on Evolutionary Computation, 1997. 1: p. 179-188.

[17]. Fulya Altiparmak Berna Dengiz and Alice E. Smith, Reliability Optimization of Computer Communication Networks Using Genetic Algorithms, 1998. p. 4676-4681.

[18]. C. A. Coello and A. D. Christiansen, Multi-objective optimization of trusses using genetic algorithms, Computers and Structures, 2000. 75(6): p. 647-660.

[19]. K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, A fast and elitist multi objective genetic algorithm, NSGA-II. IEEE Transactions on Evolutionary Computation, 2002. 6(2): p. 182-198.

[20]. H. A. Taboada, J. F. Espiritu and J. F. Coit, MOMS-GA: a multi-objective multi-state genetic algorithm for system reliability optimization design problems, IEEE Transactions on Reliability, 2008. 57(1): p. 182-191.

[21]. A. Azaron, C. Perkgoz, H. Katagiri, K. Kato and M. Sakawa, Multi-objective reliability optimization for dissimilar-unit cold-stand by systems using a genetic algorithm, Computers and Operations Research, 2009. 36(5): p. 1562-1571.

[22]. M. R. Hassan and H. Abdou, Multi-objective components assignment problem subject to lead-time constraint, Indian Journal of Science and Technology, 2018. 11(21): p. 1-9.

[23]. Ahmed Y. Hamed, Monagi H. Alkinani and M. R. Hassan, A Genetic Algorithm to Solve Capacity Assignment Problem in a Flow Network, Computers, Materials & Continua, 2020. 64(3): p. 1579-1586.

[24]. Mitsuo Gen and Runwei Cheng, Genetic Algorithms and Engineering Optimization, Wiley Series in Engineering Design and Automation, 2000.