American Journal of Engineering and Applied Sciences

Original Research Paper

Solving the Robust Design Problem for a Two-Commodity Flow Network with Node Failure

Noha Hamdy Radwan, Moatamad Refaat Hassan and Mohamed Eid Hussein

Department of Computer Science and Mathematics, Faculty of Science, Aswan University, Aswan, Egypt

Article history
Received: 13-10-2020
Revised: 10-12-2020
Accepted: 28-12-2020

Corresponding Author:
Moatamad Refaat Hassan
Department of Computer Science and Mathematics, Faculty of Science, Aswan University, Aswan, Egypt
Email: mr.hassan@sci.aswu.edu.eg

Abstract: The robust design in a flow network is one of the most important problems. It is defined as searching the optimal capacity that can be assigned to the nodes such that the network still survived even under the node's failure. This problem is considered NP-hard. So, this study presents a genetic-based algorithm to determine the maximum node capacity for a two-commodity flow network with node failure. I.e., searching the minimum sum of the assigned capacities and the maximum network reliability. The obtained results show that The proposed GA-based algorithm succeeded to solve the robust problem for the two-commodity flow network considering the node's failure.

Keywords: Flow Network, Robust Design, Two-Commodity, Node Failure, Genetic Algorithm

Introduction

Reliability evaluation is important to arrange the scheduling in many network systems (Zuo *et al*., 2007). In the case of an existing 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) (Younes and Hassan, 2011; Lin *et al*., 1995; Lin, 2001). Lin (2001) presented a simple algorithm for generating all lower boundary points to calculate system reliability. Lin (2007a) constructed a stochastic-flow network to model the information system. Lin (2007b) 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. Satitsatian and Kapur, (2006) proposed an algorithm to search for lower boundary points and used it to account for the exact reliability.

Lin (2002) Presented an algorithm to evaluate the reliability in terms of minimal paths for a two-commodity (d_1, d_2) by generating all lower boundary points for a stochastic-flow network in which each arc and node has several capacities and may fail. Hassan and Abdou, (2020) presented a method to calculate the maximum value of the demand ($d_{\max}$). The simple algorithm used to find ($d_{\max}$), its idea is based on cut sets.

In (Chen and Lin, 2008) the research debated the capacity assignment problem and offered a definition to the critical node. Prove that if the critical node fails then the system reliability drops to zero. In general, the Capacity Assignment Problem (CAP) is defined as the search for the optimal capacities that minimize the lateness in the network and maximize its reliability (Zantuti, 2008). Study structural analysis is very important to determine the critical nodes to avoid the system reliability don`t drop to zero (Chen and Lin, 2010). Structural analysis is used to identify the node`s structural and locate critical nodes to avoid the system reliability don`t drop to zero (Chen and Lin, 2010).

The robust design is defined as an issue in quality engineering where the designed product functions well even in a versatile environment. However, in the context of a stochastic-flow network, it means the network functions well even in a node`s failure situation (Chen and Lin, 2010). The robustness of an assignment is therefore recognized as the fact that the critical nodes generated after the assignment are equal to the important structural nodes (Chen and Lin, 2010; Niu *et al*, 2020). 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 (Chen, 2012).

Genetic Algorithms (GAs) are an engaging tool to solve optimization problems (Back and Schwefel, 1996). The genetic algorithm will use a search technique to find optimized or tuned or approximate solutions and Genetic algorithms are an important technique in searching for the optimum option from a set of solutions available for a particular design, (Gen and Chen, 2000; Gong *et al*., 2016; Azaron *et al*., 2009).

Younes and Hassan, (2011) 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 reach the best results in reliability optimization, to solve reliability optimization problems, many based GA approaches were proposed such as (Chen, 2012; Deeter and Smith, 1997; Dengiz *et al*., 1997). Also, GA is used to solve multiple objective optimization problems (Altiparmak *et al*., 1998; Coello and Christiansen, 2000). 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 (Hamed *et al.*, 2020).

In this study, we extend the robust design problem to a flow network with multiple demands, (Lin, 2009). Also, we propose an approach based on GA to solve the presented problem. The main objective of the proposed GA is to search for the best-assigned capacities such that the network reliability is maximized. I.e., maximize the reliability and minimize the total assigned capacities.

We used the GAs for their characteristics that distinguish them from the rest of the algorithms. GA is based on the concept of natural selection and natural genetics, these are easy to understand and enforcement, these are widely used in optimization problems, these work on a population of points rather than an individual point, these use probabilistic transition rule instead of deterministic rules, GAs can potent deal with a large number of variables. These are more effective for complex problems than simple problems (Sharma, 2013).

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

Notations and Assumptions

Notations

np	No. of paths
mp_j	Minimal paths no. j, $j = 1,2,\ldots np$
w_i^k	The weight of commodity ($k = 1,2$)
d_1	The demand of commodity 1
d_2	The demand of commodity 2
d	The demand, $d = \left(\left(w_i^1 * d_1\right) + \left(w_i^2 * d_2\right)\right)$
neq	No. of components (nodes)
n	The node
$F1$	Flow vector; $F1 = \left(f_1^1, f_2^1, \ldots, f_{np}^1\right)$
$F2$	Flow vector; $F2 = \left(f_1^2, f_2^2, \ldots, f_{np}^2\right)$
X	Capacity vector; $X = (x_1, x_2, \ldots, x_{neq})$
r	The probability
R_{d_1,d_2}	System reliability to the given demands; d_1, d_2
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

Assumptions

The following assumptions should be satisfied for the given SFN:

- Two types of commodity are transmitted from source to sink
- Capacities of different components are statistically independent
- The capacity of each node is an integer
- Flow of each commodity must satisfy the flow-conservation law (Fulkerson and Ford, 1962)

Problem Description

The Reliability Evaluation to Two Commodities

Given the demand d_1, d_2, the system reliability R_{d_1,d_2} is defined by, (Lin, 2002):

$$R_{d_1,d_2} = \Pr\left\{X \mid V(X) \geq \left(d_1, d_2\right)\right\} \tag{1}$$

where, X is a lower boundary point for d_1, d_2.

And X can be deduced from $F1 = \left(f_1^1, f_2^1, \ldots, f_{np}^1\right)$ and $F2 = \left(f_1^2, f_2^2, \ldots, f_{np}^2\right)$ by using the following equation:

$$x_i = \left\{\left[\sum_{j=1}^{np}\left(w_i^1 f_j^1 + w_i^2 f_j^2\right)\right] \mid n_i \in mp_j\right\} \quad \textit{for each}\, i = 1,2,\ldots,neq \tag{2}$$

when F generate by equations that satisfies the flowing two constraints:

$$\left\{\left[\sum_{j=1}^{np}\left(w_i^1 f_j^1 + w_i^2 f_j^2\right)\right] \mid n_i \in mp_j\right\} \leq M^1 \quad \textit{for}\ i = 1,2,\ldots,neq \tag{3}$$

$$\sum_{j=1}^{np} f_j^k = d_k \quad (k = 1,2) \tag{4}$$

The Capacity Assignment

The network structure is important to assignment the capacity. To analyze the network structure, we determine the minimum paths. The minimum path is an ordered sequence of nodes without a cycle. From the paths we specify the node covering, which extends to the definition of Structural Impact Factor (SIF) S_i for node n_i.

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 passes through n_j when no flow passes through n_i, (Chen and Lin, 2010; Niu *et al.*, 2020; Chen, 2012).

Critical Node

When the capacity node $n_i \le d$ and $R_{d_1,d_2,i}$ is zero then this node is critical node (Chen and Lin, 2010; Chen, 2012).

Probabilities Calculation Planner for Each Node

At first, we must be defining the probability for $\Pr\{x_i\}$ of node n_i to calculate $R_{d_1,d_2,i}$. To produce the corresponding capacities, we suppose find c_i components for nodes n_i. Each component has the reliability of r_i, (Chen and Lin, 2010). Then the probability for the current capacity x_i is $\Pr\{x_i\}$ and is denoted as:

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

Problem Formulation

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

$$\textit{Minimize } \mathbb{C} \tag{6}$$

$$\begin{aligned}&S.t.\\ &R_{d_1,d_2} \textit{ is maximized}\end{aligned} \tag{7}$$

where, $\mathbb{C} = \sum_i^n M^i$ and R_{d_1,d_2} is reliability corresponding to the assigned capacities under demand $d = (w_i^1 d_1 + w_i^2 d_2)$. M^i ranges from 1 to d, except for critical nodes (Chen and Lin, 2008), $M^i = d$.

The GA Components

The following subsections depict different components of the proposed GA.

Representation

A chromosome representation is needed to describe each chromosome in the GA algorithm. The chromosome M is exemplifying by a series of length (neq), where (neq) is the number of nodes as shown in this equation ($M^1, M^2,\ldots,M^{req}$).

Initial Population

The generation of an initial population is the first step in a GA. Each chromosome in generation encodes a possible solution to a problem. After creating the initial population, each chromosome is evaluated and assigned a fitness value according to the fitness function, (Diaz-Gomez and Hougen, 2007).

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)/sun_fit
End for
```

Selection

Selection process picks the fittest solutions by giving it a high percentage when picking 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 $\mu \in [0,1]$
For i = 1 to NP
        If $\mu$ >cumsum(i)
         Parent1 = i-1.
        End if
End for i
Generate a random number $\mu \in [0,1]$
For i = 1 to NP
        If $\mu$ > cumsum(i)
         Parent2 = i-1.
        End if
  End for i
End
```

Crossover

Crossover process (Lim *et al.*, 2017) is vital in generating new chromosomes by selecting two parents called parent 1 and parent 2. During crossover the parent chromosomes are taken in pairs and exchange information between two parents and generate offspring 1 and offspring 2. These offspring become next generation parent chromosomes, (Varun Kumar and Panneerselvam, 2017).

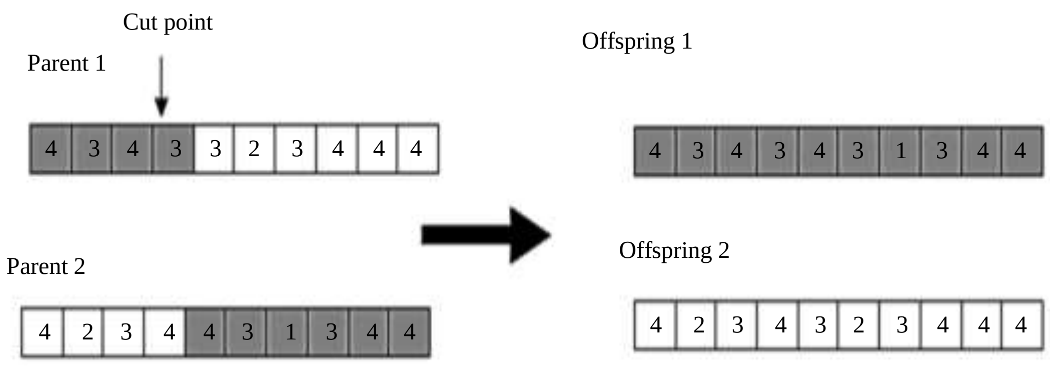

Fig. 1: Crossover process

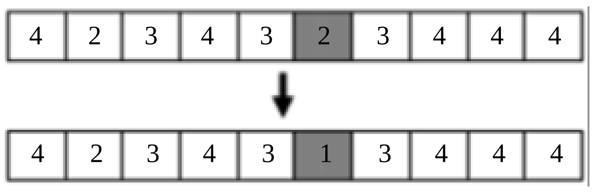

Fig. 2: Mutation process

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 Fig. 1.

Mutation

Mutation takes place after crossover is done. Mutation used to modify the genes of a chromosome selected with a mutation probability *mr*. It takes place the changes randomly to one "genes" to generate a new offspring. In this research, we will use Creep Mutation as shown in Fig. 2. 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, (Soni and Kumar, 2014).

The Proposed GA Algorithm

```
Start
Input $d_1$, $d_2$, $w_i^k$, neq, MPs, $r_i$,
and components' informations.
then compute the demand d
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
    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
```

Studied Cases

In the following subsections, the proposed GA has been applied to four 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.

Four-Node Network

This network has four nodes as shown in Fig. 3. The network has four minimal paths as follows: $mp_1 = \{n_1, n_2, n_3, n_4\}$, $mp_2 = \{n_1, n_2, n_3, n_4\}$, $mp_3 = \{n_1, n_2, n_3, n_4\}$, $mp_4 = \{n_1, n_2, n_3, n_4\}$. The nodes probability, r_i = (0.99, 0.98, 0.98, 0.99), $w_i^1 = 1$, $w_i^1 = 1.5$, Table 1 shows the best candidate M with the corresponding $\mathbb{C}$, $R_{3,2}$, execution time, Table 2 shows the best candidate M with the corresponding $\mathbb{C}$, $R_{3,2}$, execution time.

Eight-Node Network

This network has eight nodes as shown in Fig. 4. The network has eight minimal paths as follows: $mp_1 = \{n_1, n_2, n_4, n_8\}$, $mp_2 = \{n_1, n_2, n_5, n_8\}$, $mp_3 = \{n_1, n_2, n_6, n_8\}$, $mp_4 = \{n_1, n_2, n_7, n_8\}$, $mp_5 = \{n_1, n_3, n_4, n_8\}$, $mp_6 = \{n_1, n_3, n_5, n_8\}$, $mp_7 = \{n_1, n_2, n_6, n_8\}$, $mp_8 = \{n_1, n_2, n_7, n_8\}$. The nodes' probabilities, r_i = (0.90, 0.88, 0.93, 0.91, 0.89, 0.90, 0.87, 0.92). $w_i^1 = 1$, $w_i^1 = 1.5$. Table 3 shows the best candidate M with the corresponding $\mathbb{C}$, $R_{1,2}$, execution time.

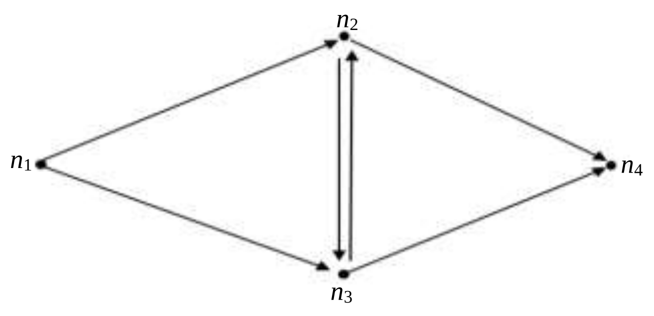

Fig. 3: Four-node network

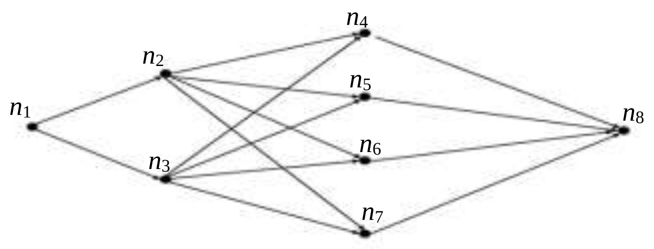

Fig. 4: Eight-node network

Table 1: The results for four-node network with $d_1 = 3$, $d_2 = 2$, $w_i^1 = 1$, $w_i^2 = 1.5$

Run no.	The best M	Sum (M)	Reliability	Execution time (in seconds)
1	[6,4,4,6]	20	0.886015	0.05
2	[6,3,4,6]	19	0.879420	0.04
3	[6,3,3,6]	18*	0.785197	0.04

*The minimum $\mathbb{C}$ found

Table 2: The results for four-node network with $d_1 = 2$, $d_2 = 3$, $w_i^1 = 1$, $w_i^2 = 1.5$

Run no.	The best M	Sum (M)	R (value)	Execution time (in seconds)
1	[7,3,4,7]	21*	0.754180	0.05
2	[7,4,4,7]	22	0.859764	0.05
3	[7,4,3,7]	21	0.754180	0.05

*The minimum $\mathbb{C}$ found

Table 3: The results for eight-node network with $d_1 = 1$, $d_2 = 2$, $w_i^1 = 1$, $w_i^2 = 1.5$

Run no.	The best M	Sum (M)	Reliability	Execution time (in seconds)
1	[4,4,4,2,3,3,2,4]	26	0.431576	0.10
2	[4,1,4,4,3,2,4,4]	26	0.444284	0.07
3	[4,1,3,3,4,2,4,4]	25	0.332267	0.04
4	[4,2,4,4,4,4,1,4]	27	0.455874	0.07
5	[4,4,4,1,4,3,2,4]	26	0.461148	0.05
6	[4,3,4,3,1,3,4,4]	26	0.463200	0.08
7	[4,3,3,1,4,4,4,4]	27	0.439850	0.07
8	[4,4,3,4,1,3,1,4]	24*	0.457769	0.08
9	[4,1,3,4,2,4,3,4]	25	0.332445	0.08
10	[4,3,2,4,1,3,4,4]	25	0.318427	0.07

*The minimum $\mathbb{C}$ found

Ten-Node Network

This network has ten nodes as shown in Fig. 5, it has seven minimal paths as follows (Chen and Lin, 2010): $mp_1 = \{n_1, n_2, n_4, n_8, n_{10}\}$, $mp_2 = \{n_1, n_2, n_5, n_8, n_{10}\}$, $mp_3 = \{n_1, n_2, n_6, n_8, n_{10}\}$, $mp_4 = \{n_1, n_2, n_6, n_9, n_{10}\}$, $mp_5 = \{n_1, n_3, n_6, n_8, n_{10}\}$, $mp_6 = \{n_1, n_3, n_6, n_9, n_{10}\}$, $mp_7 = \{n_1, n_3, n_7, n_9, n_{10}\}$. The nodes' probabilities, r_i = (0.99, 0.98, 0.97, 0.98, 0.98, 0.99, 0.97, 0.98, 0.97, 0.99), w_i^1 = 1, w_i^1 = 1.5. Table 4 shows the best candidate M with the corresponding $\mathbb{C}$, $R_{1,2}$, execution time.

Thirteen-Node Network

The network has 30 nodes as shown in Fig. 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_{14}, 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). Table 5 shows the best candidate M with the corresponding $\mathbb{C}$, $R_{1,2}$, execution time. Table 6 shows the best candidate M with the corresponding $\mathbb{C}$, $R_{1,1}$, execution time.

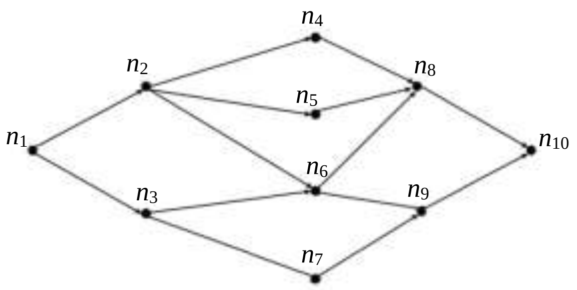

Fig. 5: Ten-node network

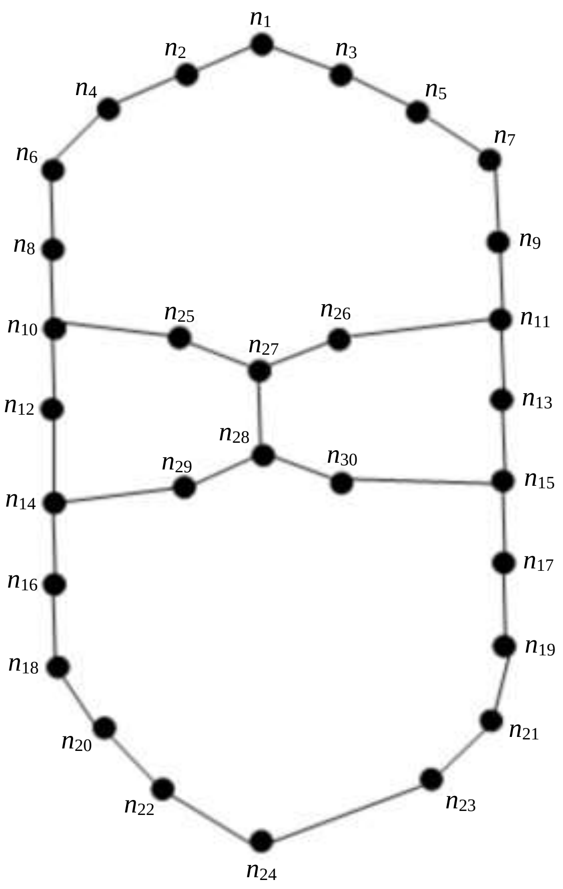

Fig. 6: Thirty-node network

Table 4: The results for ten-node network with $d_1 = 1$, $d_2 = 2$, $w_i^1 = 1$, $w_i^2 = 1.5$

Run no.	The best M	Sum (M)	Reliability	Execution time (in seconds)
1	[4,3,2,1,4,4,1,4,3,4]	30	0.867441	0.05
2	[4,4,3,1,2,4,4,3,3,4]	32	0.917677	0.07
3	[4,1,4,3,2,3,3,4,3,4]	31	0.910575	0.08
4	[4,4,4,1,4,1,4,4,3,4]	33	0.922117	0.04
5	[4,4,1,4,2,1,4,4,3,4]	31	0.914196	0.07
6	[4,2,4,2,3,4,1,1,4,4]	29*	0.910521	0.04
7	[4,2,4,1,4,4,1,4,3,4]	31	0.917152	0.07
8	[4,2,4,2,2,4,2,2,4,4]	30	0.912575	0.04
9	[4,4,1,4,3,1,1,4,4,4]	30	0.916173	0.04
10	[4,2,4,2,3,4,4,4,1,4]	32	0.912874	0.07

*The minimum $\mathbb{C}$ found

Table 5: The results for Fig. 6 network with $d_1 = 2$, $d_2 = 2$, $w_i^1 = 1$, $w_i^2 = 1$

Run no.	The best M	Sum (M)	Reliability	Execution time (in seconds)
1	[4,4,3,3,4,2,2,4,4,4,3,1,4,3,3,2,2,4,4,4,2,3,4,4,2,2,2,1,4,2]	90	0.218391	0.05
2	[4,4,4,3,3,3,2,3,3,4,1,2,4,3,3,3,4,3,4,4,2,4,4,4,1,2,4,3,3,1]	92	0.166409	0.05
3	[4,4,3,3,3,3,4,1,4,2,4,4,4,4,3,4,4,3,2,4,4,2,3,4,3,2,1,3,4,2]	95	0.204131	0.05
4	[4,3,4,4,4,2,4,4,3,4,3,2,4,2,4,3,3,2,4,4,3,1,3,4,3,2,2,2,2,4]	93	0.216591	0.05
5	[4,2,4,4,4,3,4,4,4,2,4,1,3,4,3,2,3,4,2,3,4,4,4,4,1,1,3,2,3,4]	94	0.446587	0.06
6	[4,4,2,2,4,2,4,3,3,4,3,1,1,3,4,2,2,3,4,3,3,3,2,4,2,4,4,3,2,3]	88	0.194756	0.05
7	[4,1,3,2,3,3,3,2,3,1,4,1,4,3,3,4,4,4,4,3,3,3,2,4,1,4,3,3,4,4]	90	0.140628	0.05
8	[4,4,2,2,4,3,3,2,4,2,4,2,1,4,3,4,4,4,4,4,3,3,2,4,3,1,4,2,4,4]	94	0.259161	0.06
9	[4,3,3,4,3,2,3,4,4,4,4,2,1,3,2,3,3,3,4,4,1,4,2,4,2,2,4,4,4,4]	94	0.220534	0.05
10	[4,2,3,1,4,4,4,3,3,3,4,1,4,3,3,3,4,2,3,3,4,1,3,4,1,2,2,3,4,1]	86*	0.101526	0.05

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

Table 6: The results for Fig. 6 network with $d_1 = 1$, $d_2 = 1$, $w_i^1 = 1$, $w_i^2 = 1.5$

Run no.	The best M	Sum (M)	Reliability	Execution time (in seconds)
1	[3,4,4,4,3,4,3,4,4,4,3,4,4,2,3,3,2,2,3,3,2,1,3,3,2,1,3,2,2,2]	87	0.469482	0.05
2	[3,4,3,2,2,4,4,3,2,4,3,3,4,4,4,4,3,2,2,3,3,4,3,3,1,3,2,3,4,4]	93	0.740452	0.05
3	[3,3,4,4,4,3,2,2,1,4,2,4,4,2,3,3,4,3,1,3,1,3,4,3,4,4,4,1,1,2]	86	0.358579	0.05
4	[3,3,4,4,3,2,2,1,4,1,2,4,4,2,3,4,3,3,4,3,3,3,4,3,4,1,2,1,4,1]	85	0.429503	0.05
5	[3,2,3,3,4,4,1,4,4,3,3,4,1,3,3,4,3,2,1,4,3,3,2,3,4,1,4,1,4,2]	86	0.406832	0.06
6	[3,4,3,4,2,4,2,4,4,3,2,1,4,4,3,2,4,2,4,2,4,2,2,3,4,4,4,4,3,2]	93	0.688758	0.05
7	[3,2,3,2,4,4,4,3,2,2,4,3,4,4,4,1,4,3,2,1,3,2,4,3,1,4,3,3,3,2]	87	0.453717	0.06
8	[3,4,3,3,1,4,4,3,1,2,4,2,2,2,4,4,3,2,1,3,3,3,3,3,1,2,2,4,1,3]	80*	0.277095	0.06
9	[3,4,3,3,4,2,3,2,4,3,3,2,4,4,1,4,1,4,1,3,3,3,3,3,2,4,3,2,4,3]	88	0.586286	0.05
10	[3,3,2,3,2,4,4,4,3,2,4,3,4,2,3,2,4,3,4,3,4,4,3,3,2,2,3,3,1,3]	90	0.764796	0.05

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

Conclusion

The research studied the robust design problem in the case of tow-commodity (d_1, d_2). Each node's capacity ranges from 1 to d, where $d = \left(w_i^1 d_1\right) + \left(w_i^2 d_2\right)$. The capacity of each critical node equals to d to avoid the reliability to drop to zero. The proposed GA-based algorithm succeeded to solve the robust design problem for a flow network with multiple demands. The obtained solution is distinguished by the minimum sum of the assigned capacities and the maximum reliability value.

Acknowledgement

The authors thank the anonymous referees for their careful readings and provisions of helpful suggestions to improve the presentation.

Author's Contributions

All authors are equally contributed in this work and the article.

Ethics

Authors confirm that this manuscript has not been published elsewhere and that no ethical issues are involved.

References

Altiparmak, F., Dengiz, B., & Smith, A. E. (1998, October). Reliability optimization of computer communication networks using genetic algorithms. In SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man and Cybernetics (Cat. No. 98CH36218) (Vol. 5, pp. 4676-4681). IEEE.

Azaron, A., Perkgoz, C., Katagiri, H., Kato, K., & Sakawa, M. (2009). Multi-objective reliability optimization for dissimilar-unit cold-standby systems using a genetic algorithm. Computers & Operations Research, 36(5), 1562-1571.

Back, T., & Schwefel, H. P. (1996, May). Evolutionary computation: An overview. In Proceedings of IEEE International Conference on Evolutionary Computation (pp. 20-29). IEEE.

Chen, S. G. (2012). An optimal capacity assignment for the robust design problem in capacitated flow networks. Applied Mathematical Modelling, 36(11), 5272-5282.

Chen, S. G., & Lin, Y. K. (2008). Capacity assignment for a stochastic-flow network based on structural importance. In Asian International Workshop on Advanced Reliability Modeling.

Chen, S. G., & Lin, Y. K. (2010). An approximate algorithm for the robust design in a stochastic-flow network. Communications in Statistics—Theory and Methods, 39(13), 2440-2454.

Coello, C. A., & Christiansen, A. D. (2000). Multiobjective optimization of trusses using genetic algorithms. Computers & Structures, 75(6), 647-660.

Deeter, D. L., & Smith, A. E. (1997, January). Heuristic optimization of network design considering all-terminal reliability. In Annual Reliability and Maintainability Symposium (pp. 194-199). IEEE.

Dengiz, B., Altiparmak, F., & Smith, A. E. (1997). Local search genetic algorithm for optimal design of reliable networks. IEEE transactions on Evolutionary Computation, 1(3), 179-188.

Diaz-Gomez, P. A., & Hougen, D. F. (2007, June). Initial Population for Genetic Algorithms: A Metric Approach. In Gem (pp. 43-49).

Fulkerson, D. R., & Ford, L. R. (1962). Flows in networks. Princeton University Press.

Gen, M., & Cheng, R. (2000). Genetic algorithms and engineering optimization, John Wiley&Sons. Inc., USA.

Gong, D., Sun, J., & Miao, Z. (2016). A set-based genetic algorithm for interval many-objective optimization problems. IEEE Transactions on Evolutionary Computation, 22(1), 47-60.

Hamed, A. Y., Alkinani, M. H., & Hassan, M. R. (2020). A Genetic Algorithm to Solve Capacity Assignment Problem in a Flow Network. CMC-COMPUTERS MATERIALS & CONTINUA, 64(3), 1579-1586.

Hassan, M., & Abdou, H. (2020). Cut-Set Based Method to Determine the Maximum Demand Accommodated by a Multi-State Network.

Lim, S. M., Sultan, A. B. M., Sulaiman, M. N., Mustapha, A., & Leong, K. Y. (2017). Crossover and mutation operators of genetic algorithms. International journal of machine learning and computing, 7(1), 9-12.

Lin, J. S., Jane, C. C., & Yuan, J. (1995). On reliability evaluation of a capacitated-flow network in terms of minimal pathsets. Networks, 25(3), 131-138.

Lin, Y. K. (2001). A simple algorithm for reliability evaluation of a stochastic-flow network with node failure. Computers & Operations Research, 28(13), 1277-1285.

Lin, Y. K. (2002). Two-commodity reliability evaluation for a stochastic-flow network with node failure. Computers & Operations Research, 29, 1927-1939.

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

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

Lin, Y. K. (2009). A MP-based algorithm for a multicommodity stochastic-flow network with capacity weights. International Journal of Industrial Engineering: Theory Applications and Practice, 16(4), 282-292.

Niu, Y. F., Xu, X. Z., He, C., Ding, D., & Liu, Z. Z. (2020). Capacity Reliability Calculation and Sensitivity Analysis for a Stochastic Transport Network. IEEE Access, 8, 133161-133169.

Satitsatian, S., & Kapur, K. C. (2006). An algorithm for lower reliability bounds of multistate two-terminal networks. IEEE Transactions on Reliability, 55(2), 199-206.

Sharma, M. (2013). Role and Working of Genetic Algorithm in Computer Science. International Journal of Computer Applications and Information Technology (IJCAIT), 2(1).

Soni, N., & Kumar, T. (2014). Study of various mutation operators in genetic algorithms. International Journal of Computer Science and Information Technologies, 5(3), 4519-4521.

Varun Kumar, S. G., & Panneerselvam, R. (2017). A study of crossover operators for genetic algorithms to solve VRP and its variants and new sinusoidal motion crossover operator. International Journal of Computational Intelligence Research, 13(7), 1717-173

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

Zantuti, A. F. (2008, August). Algorithms for capacities and flow assignment problem in computer networks. In 2008 19th International Conference on Systems Engineering (pp. 315-318). IEEE.

Zuo, M. J., Tian, Z., & Huang, H. Z. (2007). An efficient method for reliability evaluation of multistate networks given all minimal path vectors. IIE transactions, 39(8), 811-817.