An application of genetic algorithm in solving reliability optimization problem under interval component Weibull parameters

Laxminarayan Sahoo¹*, Asoke Kumar Bhunia² and Dilip Roy³

¹ Department of Mathematics, Raniganj Girls’ College, India
E-mail: lxsahoo@gmail.com

² Department of Mathematics, The University of Burdwan, India
E-mail: math_akbhunia@buruniv.ac.in

³ Centre for Management Studies, The University of Burdwan, India
E-mail: dr.diliproy@gmail.com

Abstract. This paper deals with the reliability optimization problem of a complicated system where time-to-failure of each component follows the Weibull distribution with imprecise parameters. In the existing work, either both the scale and shape parameters of Weibull distribution or the scale parameter as random variable with known distribution are considered as fixed. However, in reality, both the parameters may vary due to some factors and it is sensible to treat them as imprecise numbers. Here, this imprecise number is represented by an interval number. In this paper, we have formulated the reliability optimization problem with Weibull distributed time-to-failure for each component. The corresponding problem has been formulated as an unconstrained mixed-integer programming problem with interval coefficients using penalty function technique and solved by genetic algorithm. Finally, a numerical example has been solved for different types of scale and shape parameters of Weibull distribution.

Keywords: Reliability optimization, Weibull parameters, Redundancy allocation, Genetic algorithm, Interval number, penalty technique.

1 Introduction

During the last few decades, attention is being paid to reliability redundancy allocation problems which have started drawing the attention of the reliability designers for arriving at reliability optimization designs [1-4]. The basic objective of a redundancy allocation problem (RAP) is to increase the reliability of subsystems so as to arrive at a prefixed reliability goal for the system as a whole, subject to several operating constraints on the system/subsystem. RAP is basically a nonlinear integer/mixed-integer programming problem. Most of these problems cannot be solved by direct/indirect or mixed search methods due to discrete search space. According to Chern [4] RAP is NP-hard and it has been well studied as summarized in Tillman et al. [5] and Kuo and Prasad [1]. In the earlier stage, several deterministic methods, like heuristic methods [6-8], reduced gradient method [9], dynamic programming method [10], branch and bound method [2], integer programming [11], mixed-integer and nonlinear programming [5], linear programming approach [6] were used for solving such RAP. However, these methods have both advantages

Received Nov 10, 2011 / Accepted Jan 5, 2012
and disadvantages. Dynamic programming is not useful for reliability optimization of a general system as it can be used only for a few particular structures of the objective function and constraints that are decomposable. In branch and bound method, the effectiveness depends on sharpness of the bound and required memory increase exponentially with the problem size. As a result, with the advent of genetic algorithm (GA) [12-13] and other evolutionary algorithms, researchers have started paying more attention to RAP using numerical methods [3], [14-15] as these methods provide more flexibility and require less assumptions on the objective as well as the constraints and are also effective irrespective of whether the search space is discrete or not. These have enabled the reliability planners/designers to undertake and reasonably compromise with several goals.

In the literature in almost all the studies referred above, the design parameters involved in RAP has usually been taken to be precise values. This means that every probability involved is perfectly determinable. In this case, it is usually assumed that there exist some complete probabilistic information about the system and the component behavior. However, in real life situations, there are not sufficient statistical data available in most of the cases where the system is new or exists only as a project. It is not always possible to observe the stability from the statistical point of view. This means that only some partial information about the system components is known. So the reliability of a component of a system will be an imprecise number which can be represented by an interval number and is calculated by imprecise probabilities [16-18]. Further, distributional parameters may not be of precise value. They may be allowed to vary over an interval to take care of the sensitivity of several factors. Keeping these considerations in mind, the reliability optimization problem can be described as a problem with distributional parameters assuming interval/imprecise values. Recently, studies of the RAP have already been initiated where the component reliabilities are imprecise and or interval valued by some authors [19-23].

In this work, we propose to consider the RAP under imprecise reliability and component reliability following the Weibull distribution with interval valued distributional parameters. The problem is formulated as a non-linear constrained mixed-integer programming problem with interval coefficients for maximizing the overall system reliability under resource constraints. During the last few years, several techniques were proposed for solving constraints optimization problem with fixed/precise coefficient with the help of genetic algorithm. Among them, penalty function techniques are very popular in solving the same by GAs [19], [20], [24]. In this work, to solve the constrained optimization problem we have converted it into an unconstrained one by using the penalty function technique and the resulting objective function would be interval valued. So, to solve this type of problem by GA, order relations of interval numbers are essential for GA operators. In these works, we have used the definitions of Sahoo et al. [23] for order relations between interval numbers. For solving such optimization problem by GA, we have developed a real coded elitist GA with tournament selection, uniform crossover and one-neighborhood mutation for integer variables and whole arithmetical crossover and boundary mutation for floating point variables. Finally, to illustrate the proposed model, a numerical example has been solved for different cases of scale and shape parameters of Weibull distribution.
2 Assumptions and notations

In formulation of the problem, the following assumptions have been considered.
(i) The chance of failure of any component is independent.
(ii) All the redundancy is active redundancy without repair.
(iii) Failure of each component follows the Weibull distribution.
(iv) Both the Weibull scale and shape parameters are imprecise and interval valued.

To develop the paper, the following notations have been used.

Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>the number of subsystems</td>
</tr>
<tr>
<td>$x_i$</td>
<td>the number of components in $i$th subsystem, arranged in parallel</td>
</tr>
<tr>
<td>$r_i$</td>
<td>reliability of $i$th component</td>
</tr>
<tr>
<td>$t$</td>
<td>mean time-to-failure</td>
</tr>
<tr>
<td>$(x,t)$</td>
<td>$(x_1, x_2, ..., x_n, t)$</td>
</tr>
<tr>
<td>$\alpha_i = [\alpha_{IL}, \alpha_{IR}]$</td>
<td>interval valued weibull scale parameter for $i$th subsystem</td>
</tr>
<tr>
<td>$\beta_i = [\beta_{IL}, \beta_{IR}]$</td>
<td>interval valued weibull shape parameter for $i$th subsystem</td>
</tr>
<tr>
<td>$r_i(t) = [r_{IL}(t), r_{IR}(t)]$</td>
<td>$e^{-(\alpha_{IL} \cdot t \cdot \beta_{IL} + \beta_{IR} \cdot t \cdot \beta_{IR})}, i = 1, 2, \ldots, n$</td>
</tr>
<tr>
<td>$R_i(x,t) = [R_{IL}(x,t), R_{IR}(x,t)]$</td>
<td>$1 - (1 - [r_{IL}(t), r_{IR}(t)]^x)^\sum$, the reliability of $i$th parallel subsystem</td>
</tr>
<tr>
<td>$R_S(x,t) = [R_{SL}(x,t), R_{SR}(x,t)]$</td>
<td>system reliability</td>
</tr>
<tr>
<td>$\mathbb{X}$</td>
<td>set of real numbers</td>
</tr>
<tr>
<td>$X$</td>
<td>feasible region</td>
</tr>
<tr>
<td>$b_i$</td>
<td>availability of $i$th resource ($i = 1, 2, ..., m$)</td>
</tr>
<tr>
<td>$l_i, u_i$</td>
<td>lower and upper bounds of $x_i$</td>
</tr>
<tr>
<td>$l_t, u_t$</td>
<td>lower and upper bounds of $t$</td>
</tr>
<tr>
<td>$p_{\text{cross}}$</td>
<td>probability of crossover/ crossover rate</td>
</tr>
<tr>
<td>$p_{\text{mute}}$</td>
<td>probability of mutation/ mutation rate</td>
</tr>
<tr>
<td>$p_{\text{size}}$</td>
<td>population size</td>
</tr>
<tr>
<td>$\text{max}_\text{gen}$</td>
<td>maximum number of generation</td>
</tr>
</tbody>
</table>
According to the assumptions, the system reliability would be interval valued. So to optimize this system reliability under certain constraints, the following topics play important role in solving the problem by genetic algorithm.

(i) Interval mathematics
(ii) Interval order relations
(iii) Weibull distribution with interval valued parameters

Now, we shall discuss these topics in details.

3 Interval mathematics, interval order relations and Weibull distribution with interval valued parameters

An interval number $A$ is a closed interval denoted by $A = [a_L, a_R]$ and is defined by $A = [a_L, a_R] = \{ y : a_L \leq y \leq a_R, y \in \mathbb{R}^* \}$, where $a_L, a_R$ are the left and right limits respectively and $\mathbb{R}^*$ is the set of all real numbers. Actually, every real number can be treated as an interval, such as for all $y \in \mathbb{R}^*$, $y$ can be written as an interval $[y, y]$ which has zero width. Now we shall present some basic formulas of interval mathematics as follows:

Let $A = [a_L, a_R]$ and $B = [b_L, b_R]$ be two intervals.

Addition: $A + B = [a_L, a_R] + [b_L, b_R] = [a_L + b_L, a_R + b_R]$
Subtraction: $A - B = [a_L, a_R] - [b_L, b_R] = [a_L - b_R, a_R - b_L]$
Scalar Multiplication: $\lambda A = \lambda [a_L, a_R] = \begin{cases} [\lambda a_L, \lambda a_R] & \text{if } \lambda \geq 0 \\ [\lambda a_R, \lambda a_L] & \text{if } \lambda < 0 \end{cases}$
Multiplication:
$A \times B = [a_L, a_R] \times [b_L, b_R] = [\min(a_L b_L, a_L b_R, a_R b_L, a_R b_R), \max(a_L b_L, a_L b_R, a_R b_L, a_R b_R)]$
Division: $\frac{A}{B} = \frac{1}{B} [a_L, a_R] \times \left[ \frac{1}{b_R}, \frac{1}{b_L} \right]$, provided $0 \notin [b_L, b_R]$

3.1 Function of finite interval

For monotonically increasing function $f(y)$ in the interval $A = [a_L, a_R]$, where $y \in \mathbb{R}^*$ then $f(A) = f([a_L, a_R]) = f(a_L), f(a_R))$. Similarly, if $f(y)$ is monotonically decreasing in the interval $A = [a_L, a_R]$, then $f(A) = f([a_L, a_R]) = f(a_R), f(a_L))$. 

Exponential: $\text{Exp}(A) = [\text{Exp}(a_L), \text{Exp}(a_R)]$ and $\text{Exp}(-A) = [\text{Exp}(-a_R), \text{Exp}(-a_L)]$

Logarithm: $\text{log}(A) = [\text{log}(a_L), \text{log}(a_R)]$

3.2 Integration of an interval function

According to Moore [25], the integration of an interval function is defined by

$$\int_{a}^{b} f(y) \, dy = \left[ \int_{a}^{b} f_L(y) \, dy, \int_{a}^{b} f_R(y) \, dy \right] \text{ for any } y \in \mathbb{R}.$$ (1)

Here $f(y) = [f_L(y), f_R(y)]$ and both $f_L(y)$ and $f_R(y)$ are continuous real valued functions.

3.3 Integral power of an interval

According to Hansen and Walster [26], the integral power of an interval is defined by

$$A^n = [a_L^n, a_R^n] = \begin{cases} [1,1] & \text{if } n = 0 \\ [a_L^n, a_R^n] & \text{if } a_L \geq 0 \text{ or } n \text{ is odd} \\ [a_L^n, a_R^n] & \text{if } a_R \leq 0, n \text{ is even} \\ [0, \max(a_L^n, a_R^n)] & \text{if } a_L \leq 0 \leq a_R, n > 0 \text{ is even} \end{cases}$$ (2)

3.4 Complex Interval

Let $A = [a_L, a_R]$ and $B = [b_L, b_R]$. A complex interval $Z$ is identified with the interval vector $Z = A + iB = [a_L, a_R] + i[b_L, b_R]$. The basic arithmetical operations of complex intervals like, addition, subtraction, division and multiplication are the same as the real interval arithmetic available in the Book written by Baker [27].

3.5 Interval power of an interval

Sahoo et al. [23] defined the interval power of an interval as follows:

Let $A = [a_L, a_R]$ and $B = [b_L, b_R]$ be two intervals, then

$$(A)^B = [a_L^{b_L}, a_R^{b_R}]$$
Laxminarayan Sahoo, Asoke Kumar Bhunia and Dilip Roy/An application of genetic algorithm in solving reliability optimization problem under interval component Weibull parameters. MJOR Vol. 1, No. 1, Jul-Dec 2012, pp. 2-19. EDITADA

\[
\begin{align*}
&= \begin{cases} 
\left( e^{\min(b_l \log a_l, b_k \log a_l, b_k \log a_k, b_k \log a_k)} , e^{\max(b_l \log a_l, b_k \log a_l, b_k \log a_k, b_k \log a_k)} \right) & \text{if } a_l \geq 0 \\
\text{a complex interval} & \text{if } a_l < 0
\end{cases}
\end{align*}
\]

If \( A = [-a_L, -a_R] \) and \( B = [b_L, b_R] \) be two intervals, then

\[
[-a_L, -a_R] \cap [b_L, b_R] = [a_R, a_L] \cap [b_L, b_K] + \cos[(2k + 1)\pi b_L, (2k + 1)\pi b_R] + i \sin[(2k + 1)\pi b_L, (2k + 1)\pi b_R]
\]

if \( a_l, a_R \geq 0, k = 0, 1, 2, 3, \ldots \)

3.6 Interval order relations

According to the assumption (i), the objective function of redundancy allocation problem would be interval valued. So, to arriving at the optimum solution of the said problem, the order relations of interval numbers play an important role in decision making.

Let \( A = [a_L, a_R] \) and \( B = [b_L, b_R] \) be two intervals. Then these two intervals may be of the following three types:

Type-1: Two intervals are disjoint.
Type-2: Two intervals are partially overlapping.
Type-3: One of the intervals contained the other one.

It is to be noted that both the intervals \( A = [a_L, a_R] \) and \( B = [b_L, b_R] \) will be equal in case of fully overlapping intervals. That is \( A = B \) if \( a_L = b_L \) and \( a_R = b_R \).

Here we shall consider the definitions of order relations developed by Sahoo et al. [23].

Definition 3.6.1 The order relation \( >_{\max} \) between the intervals

\( A = [a_L, a_R] = \langle a_c, a_w \rangle \) and \( B = [b_L, b_R] = \langle b_c, b_w \rangle \), then for maximization problems

(i) \( A >_{\max} B \iff a_c > b_c \) for Type-1 and Type-2 intervals,
(ii) \( A >_{\max} B \iff \) either \( a_c \geq b_c \wedge a_w < b_w \) or \( a_c \geq b_c \wedge a_L > b_L \) for Type-3 intervals,

According to this definition, the interval \( A \) is accepted for maximization case. Clearly the order relation \( >_{\max} \) is reflexive and transitive but not symmetric.
Definition 3.6.2 The order relation $\leq_{\text{min}}$ between the intervals $A = [a_L, a_R] = \{a_c, a_w\}$ and $B = [b_L, b_R] = \{b_c, b_w\}$, then for minimization problems

(i) $A <_{\text{min}} B \iff a_c < b_c$ for Type-1 and Type-2 intervals,

(ii) $A <_{\text{min}} B \iff$ either $a_c \leq b_c \wedge a_w < b_w$ or $a_c \leq b_c \wedge a_l < b_l$ for Type-3 intervals,

According to this definition, the interval $A$ is accepted for minimization case. Clearly the order relation $<_{\text{min}}$ is reflexive and transitive but not symmetric.

3.7 Mean, Variance and Standard deviation of Interval Numbers

According to the Bhunia et al. [20], mean, variance and standard deviation of $n$ interval numbers are defined as follows:

Let $y_i = [y_{iL}, y_{iR}]$, $i = 1, 2, ..., n$ be the $i$th observation which is an interval number. Then mean, variance and standard deviation of $y_i = [y_{iL}, y_{iR}]$, $i = 1, 2, ..., n$ are given by

$$
\bar{y} = \left[ \frac{1}{n} \sum_{i=1}^{n} y_{iL}, \frac{1}{n} \sum_{i=1}^{n} y_{iR} \right]
$$

(3)

$$
\text{Var}(y) = \left[ \sigma^2_L, \sigma^2_R \right] = \frac{1}{n} \sum_{i=1}^{n} \left( y_{iL} - \frac{1}{n} \sum_{i=1}^{n} y_{iL}, y_{iR} - \frac{1}{n} \sum_{i=1}^{n} y_{iR} \right)^2
$$

(4)

and $\sigma_y = [\sigma_L, \sigma_R] = \sqrt{\text{Var}(y)} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left( y_{iL} - \frac{1}{n} \sum_{i=1}^{n} y_{iL}, y_{iR} - \frac{1}{n} \sum_{i=1}^{n} y_{iR} \right)^2}

(5)

3.8 Weibull distribution with interval valued parameters

The probability density function for a Weibull distributed $t$ is given by

$$
f(t) = \frac{\beta(t - \delta)^{\beta - 1}}{(\theta - \delta)^\beta} \exp \left[ -\left( \frac{t - \delta}{\theta - \delta} \right)^\beta \right], \quad t \geq \delta \geq 0
$$

(6)

Where $\beta$ is known as the shape parameter and $(\theta - \delta)$, known as the scale parameter. Both the parameters are always positive.

If $\delta = 0$ and $\theta^{-\beta} = \alpha$ then $f(t) = \alpha \beta t^{\beta - 1} \exp \left[ -\alpha t^\beta \right], \quad t \geq 0$
Now if \( \alpha = [\alpha_L, \alpha_R] \) and \( \beta = [\beta_L, \beta_R] \) then \( f(t) \) can be written as an interval \([f_L(t), f_R(t)]\) where
\[
f_L(t) = \alpha_L \beta_L t^{\beta_L - 1} \exp\left[-\alpha_L t^{\beta_L}\right] \quad \text{and} \quad f_R(t) = \alpha_R \beta_R t^{\beta_R - 1} \exp\left[-\alpha_R t^{\beta_R}\right], \quad t \geq 0.
\]

We can easily ensure from interval mathematics that for such a distribution, the following properties holds.

Property-1: \([f_L(t), f_R(t)] \geq [0, 0]\) for \( t \geq 0 \)

Property-2: \( \int_{-\infty}^{\infty} [f_L(t), f_R(t)] dt = [1, 1] \)

So, it can be easily prove that \([f_L(t), f_R(t)]\) is interval valued probability density function. The interval valued probability distribution function for a weibull distributed \( t \) is given by
\[
F(t) = [F_L(t), F_R(t)] = \left[1 - \exp(-\alpha_L t^{\beta_L}), 1 - \exp(-\alpha_R t^{\beta_R})\right]
\]
As \( r(t) = 1 - F(t) \), therefore the interval valued reliability function of interval valued Weibull distribution is given by
\[
r(t) = [r_L(t), r_R(t)] = \left[\exp(-\alpha_R t^{\beta_R}), \exp(-\alpha_L t^{\beta_L})\right]
\]
Therefore \( r_L(t) = \exp(-\alpha_R t^{\beta_R}) \) and \( r_R(t) = \exp(-\alpha_L t^{\beta_L}) \)

4 Problem formulation

Our objective is to formulate the redundancy allocation problem of a complicated system with \( n \) subsystems. The goal of the redundancy allocation problem is to determine the number of redundant components in each of \( n \) parallel subsystems and mission time for overall system so as to maximize the overall system reliability subject to the given constraints mostly arriving in linear form. The time-to-failure for each available component is distributed according to a two-parameter Weibull distribution with imprecise scale and shape parameters. Then the corresponding problem becomes a mixed-integer nonlinear programming problem with \( m \) constraints, which can be formulated as follows:
\[
\text{Maximize} \quad R_S(x, t) = f(R_1(x_1, t), R_2(x_2, t), \ldots, R_q(x_q, t), \ldots, R_n(x_n, t)) \tag{7}
\]
subject to \( Ax \leq b \tag{8} \)
\[
\text{where } A = \begin{bmatrix}
c_{11} & c_{12} & \cdots & c_{1n} \\
c_{21} & c_{22} & \cdots & c_{2n} \\
. & . & \cdots & . \\
c_{m1} & c_{m2} & \cdots & c_{mn}
\end{bmatrix} \quad \text{and } b = [b_1 \ b_2 \ \ldots \ b_m]^T
\]
5 Genetic Algorithm based constraints handling approach

In the application of GA for solving reliability optimization problem with interval objective, there arises an important question for handling the constraints relating to the problem. During the past, several methods have been proposed to handle the constraints in evolutionary algorithms [13], [24] for solving the same problem with fixed objective. These methods can be classified into several types, viz. penalty function techniques, methods that preserve the feasibility of solutions, methods that clearly distinguish between feasible and infeasible solutions and hybrid methods. Among these methods, penalty function technique is very well known and widely applicable. In this technique, the amount of constraint violations is added/subtracted to the objective function in different ways. To overcome this situation, we have to solve the same problem with an interval-valued fitness function by penalizing a large positive number (say $M$ which can be written in the interval form $[M, M]$) is assigned to the interval objective function for the infeasible solution of the problem [19-20]. We call this technique as Big-M penalty technique. In this case, the transformed problem is as follows:

$$\text{Maximize } R(x, t) = R(x, t) + \delta(x, t)$$

where $\delta(x, t) = \begin{cases} [0, 0] & \text{if } (x, t) \in X \\ -R(x, t) - [M, M] & \text{if } (x, t) \notin X \end{cases}$

and $X = \{(x, t) : Ax \leq b, 1 \leq l_i \leq x_i \leq u_i \text{ and } t_i \leq t \leq t_u \}$ be the feasible space.

This problem (9) is a mixed-integer non-linear unconstrained optimization problem with interval objective of $n$ integer variables $x_1, x_2, ..., x_n$ and a single floating point variable $t$. For solving this problem, we have developed a real coded Genetic Algorithm (GA) with advanced operators for mixed integer variables.

Genetic Algorithm is a well-known probabilistic method for solving an optimization problem. A more detailed implementation of genetic algorithm was given by Goldberg [12]. Genetic algorithm is a part of evolutionary computing, which is a vast growing area of artificial intelligence. Inspired by theory of evolution “The survival of relatively fit”- GA is computer algorithm, which create an environment where populations of data can compete and
only the fittest survive. It has successfully been applied in different real world application problems. This algorithm starts with an initial population of chromosomes. These populations will be improved from generation to generation by an artificial evolution process. During each generation, each chromosome in the entire population is evaluated using the measure of fitness and the population of the next generation is created through different genetic operators. This algorithm can be implemented easily with the help of computer programming. In particular, it is very useful for solving complex optimization problems which cannot be solved easily by analytical /direct/gradient based mathematical techniques.

For implementing the GA in solving the optimization problems, the following basic components are to be considered.

- **GA Parameters**
- **Chromosome representation**
- **Initialization of population**
- **Evaluation of fitness function**
- **Selection/ reproduction process**
- **Genetic operators (crossover, mutation and elitism)**
- **Termination criteria**

There are four basic parameters of GA, viz. \( p_{\text{size}} \) (population size i.e., the number of individuals in the population), \( \text{max}_\text{gen} \) (maximum number of generations/iterations), \( p_{\text{cross}} \) (probability of crossover) and \( p_{\text{mute}} \) (probability of mutation). There is no hard and fast rule for the choice of first two parameters i.e., \( p_{\text{size}} \) and \( \text{max}_\text{gen} \). These vary from problem to problem according to the dimension. \( p_{\text{cross}} \) is the value, how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parent’s chromosome. \( p_{\text{mute}} \) is the value, how often parts of chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover without any change. If mutation is performed, one or more parts of chromosome are changed. Again, from the natural genetics, it is obvious that the value of \( p_{\text{cross}} \) should be greater than \( t \) that of the value of \( p_{\text{mute}} \). Generally, the crossover probability varies from 0.8 to 0.95 whereas the mutation probability varies from 0.05 to 0.20.
Initially, the chromosomes/individuals are generated randomly. In this work, each chromosome/individual has \((n+1)\) components/genes of which first \(n\) genes are relating to integer variables whereas the last one is relating to floating point variable. These chromosomes/individuals compete with each other with their fitness values. Here, the transformed unconstrained objective function due to Big-M penalty is considered as the fitness function. In the proposed GA, the well known tournament selection process is employed for selection/ reproduction. The primary objective of this process is to emphasize the above average solutions and eliminate the below average solutions from
the population for the next generation under the well-known evolutionary principle “survival of the fittest”. This selection procedure is based on the following assumptions:

(i) When both the chromosomes/individuals are feasible then the one with better fitness value is selected.
(ii) When one chromosome/individual is feasible and another is infeasible then the feasible one is selected.
(iii) When both the chromosomes/individuals are infeasible with unequal constraint violations, then the chromosome with less constraint violation is selected.
(iv) When both the chromosomes/individuals are infeasible with equal constraint violations, then any one chromosome/individual is selected.

After the selection process, new offspring will be created through crossover and mutation processes. Crossover is considered to be the main search operator and is used to thoroughly explore the search process. In crossover process the genetic information between two or more chromosomes/individuals are blended to produce new individuals. After a crossover is performed, mutation process takes place. Mutation processes play an important role in genetic algorithm. Mutation is intended to prevent to the falling of all solutions in the population into a local optimum of the solved problem. Mutation operation randomly changes the offsprings resulted from crossover. In this work, we have used uniform crossover and one-neighborhood mutation in the genes corresponding to the integer variables, whole arithmetical crossover and boundary mutation for the last gene of the chromosome. The computational steps of crossover are as follows:

**Step-1:** Find the integral value of \( \lfloor \text{p\_cross} \times \text{p\_size} \rfloor \) and store it in \( N \).

**Step-2:** Select two chromosomes \( V_k \) and \( V_i \) randomly from the population.

**Step-3:** For first \( n \) genes, compute the components \( \bar{x}_{ij} \) and \( \bar{x}_{ij} \) \( (j = 1, 2, \ldots, n) \) of two offspring by either \( \bar{x}_{ij} = x_{ij} - g \) and \( \bar{x}_{ij} = x_{ij} + g \) if \( x_{kj} > x_{ij} \)
or, \( \bar{x}_{ij} = x_{kj} + g \) and \( \bar{x}_{ij} = x_{ij} - g \), where \( g \) is a random integer number between 0 and \( |x_{kj} - x_{ij}| \), \( j = 1, 2, \ldots, n \)

and for the last gene, compute the last components \( x'_{kj} \) and \( x'_{ij} \) of two offspring will be created by

\[
x'_{kj} = c x_{kj} + (1-c) x_{ij} \quad \text{and} \quad x'_{ij} = (1-c) x_{kj} + c x_{ij}
\]

where \( c \) is a random number between 0 and 1.
Step-4: Repeat Step-2 and step-3 for \( \frac{N}{2} \) times.

The computational steps of mutation are as follows:

Step-1: Find the integral value of \( p_{\text{size}} \times p_{\text{mute}} \) and store it in \( N \).

Step-2: Select a chromosome \( V_i \) randomly from the population.

Step-3: Select a particular gene \( V_{ik} (k=1,2,...,n,n+1) \) of chromosome \( V_i \) for mutation and domain of \( V_{ik} \) is \([l_{ik},u_{ik}]\).

Step-4: Create new gene \( V_{ik}' \) corresponding to the selected gene \( V_{ik} \) by mutation process as follows:

For \( k=1,2,...,n \)

\[
V_{ik}' = \begin{cases} 
  v_{ik} + 1 & \text{if } v_{ik} = l_{ik} \\
  v_{ik} - 1 & \text{if } v_{ik} = u_{ik} \\
  v_{ik} + 1 & \text{if a random digit is 0} \\
  v_{ik} - 1 & \text{if a random digit is 1}
\end{cases}
\]

Otherwise

\[
V_{ik}' = \begin{cases} 
  l_{ik} & \text{if a random digit is 0} \\
  u_{ik} & \text{if a random digit is 1}
\end{cases}
\]

Step-5: Repeat Step-2 to Step-4 for \( N \) times.

Sometimes, in any generation, there is a chance that the best chromosome may be lost when a new population is created by crossover and mutation operations. To remove this situation the worst individual/chromosome is replaced by that best individual/chromosome in the current generation. This process is called elitism.

The termination condition is a condition for which the algorithm/process is going to stop. For this purpose any one of the following three conditions is considered as the termination conditions.

(i) the best individual does not improve over specified generations,

(ii) the total improvement of the last certain number of best solutions is less than a pre-assigned small positive number, and

(iii) the number of generations reaches a prescribed finite number of generation (called maximum number of generations).

The pictorial representation of the GA process is shown in Fig.1.
6 Numerical Example

To study the performance of the Genetic Algorithm for solving reliability optimization problem for a complex system, a numerical example of five-link bridge network system has been considered (see Fig. 2). The proposed method is coded in C programming language and runs in the Linux operating system. The computational procedure has been done on PC with Intel core-2 duo processor with 3.0GHz. For each case, 50 independent runs have been performed to calculate the best found system reliability which is nothing but the optimal value of system reliability), mean, standard deviation of system reliability in interval forms and average CPU time. In this computational work, the values of different genetic parameters like, population size \( p\text{-}\text{size} \), mutation rate \( p\text{-}\text{mute} \), crossover rate \( p\text{-}\text{cross} \) and maximum generation \( \text{max}\_\text{gen} \) have been taken as 200, 0.15, 0.85 and 150 respectively.

Fig. 2. Five-link bridge network system.
For the five-link bridge system, the optimization problem is given by

Maximize  

\[ R_5(x, t) = R_1R_2 + Q_2R_3R_4 + Q_1R_2R_3R_4 + R_1Q_2Q_3R_4R_5 + Q_1R_2R_3Q_4R_5 \]

subject to

\[ c(x) = x_1 + 0.1x_2 + 2.0x_3 + x_4 + x_5 \leq 25 \]

\[ w(x) = 2x_1 + x_2 + x_3 + 0.1x_4 + x_5 \leq 21 \]

\[ v(x) = x_1 + x_2 + x_3 + x_4 + x_5 \leq 28 \]

where  

\[ R_i = 1 - (1 - r_i(t))^{x_i}, i = 1, 2, 3, 4, 5 \text{ and } Q_j = 1 - R_j, j = 1, 2, 3, 4, 5. \]

\[ r_i(t) = e^{-\alpha_i t^{\beta_i}}, i = 1, 2, 3, 4, 5 \]

1 \leq x_i \leq 6, x_i \text{ being integer, } i = 1, 2, 3, 4, 5

1 \leq t \leq 5, t \text{ being real valued}

To study the variation of the parameters, we consider the following four cases as follows:

Case-I: When both the parameters  \( \alpha_i \) and  \( \beta_i \) are interval valued numbers

Case-II: When  \( \alpha_i \) ’s are interval valued and  \( \beta_i \) ’s are fixed valued numbers

Case-III: When  \( \alpha_i \) ’s are fixed valued and  \( \beta_i \) ’s are interval valued numbers

Case-IV: When both the parameters  \( \alpha_i \) and  \( \beta_i \) are fixed valued numbers

The values of  \( \alpha_i \) and  \( \beta_i \) ( \( i = 1, 2, 3, 4, 5 \) ) are given in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>Case-I</th>
<th>Case-II</th>
<th>Case-III</th>
<th>Case-IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>([\alpha_1L, \alpha_1R])</td>
<td>([0.257,0.258])</td>
<td>([0.257,0.258])</td>
<td>([0.257,0.258])</td>
</tr>
<tr>
<td>2</td>
<td>([\alpha_2L, \alpha_2R])</td>
<td>([0.118,0.119])</td>
<td>([0.118,0.119])</td>
<td>([0.118,0.118])</td>
</tr>
<tr>
<td>3</td>
<td>([\alpha_3L, \alpha_3R])</td>
<td>([0.214,0.215])</td>
<td>([0.214,0.215])</td>
<td>([0.214,0.214])</td>
</tr>
<tr>
<td>4</td>
<td>([\alpha_4L, \alpha_4R])</td>
<td>([0.165,0.166])</td>
<td>([0.165,0.166])</td>
<td>([0.165,0.165])</td>
</tr>
<tr>
<td>5</td>
<td>([\alpha_5L, \alpha_5R])</td>
<td>([0.210,0.211])</td>
<td>([0.210,0.211])</td>
<td>([0.210,0.210])</td>
</tr>
<tr>
<td>6</td>
<td>([\beta_1L, \beta_1R])</td>
<td>([1.99,2.1])</td>
<td>([1.99,1.99])</td>
<td>([1.99,2.1])</td>
</tr>
<tr>
<td>7</td>
<td>([\beta_2L, \beta_2R])</td>
<td>([1.99,2.1])</td>
<td>([1.99,1.99])</td>
<td>([1.99,2.1])</td>
</tr>
<tr>
<td>8</td>
<td>([\beta_3L, \beta_3R])</td>
<td>([1.99,2.1])</td>
<td>([1.99,1.99])</td>
<td>([1.99,2.1])</td>
</tr>
<tr>
<td>9</td>
<td>([\beta_4L, \beta_4R])</td>
<td>([1.99,2.1])</td>
<td>([1.99,1.99])</td>
<td>([1.99,2.1])</td>
</tr>
<tr>
<td>10</td>
<td>([\beta_5L, \beta_5R])</td>
<td>([1.99,2.1])</td>
<td>([1.99,1.99])</td>
<td>([1.99,2.1])</td>
</tr>
</tbody>
</table>

Table 1. Shows the values of  \( \alpha_i \) and  \( \beta_i \) ( \( i = 1, 2, 3, 4, 5 \) ) for four different cases.
Laxminarayan Sahoo, Asoke Kumar Bhunia and Dilip Roy/ An application of genetic algorithm in solving reliability optimization problem under interval component Weibull parameters. MJOR Vol. 1, No. 1, Jul-Dec 2012, pp. 2-19. EDITADA

The solutions of five-link bridge network system for different cases have been displayed in Table 2. From Table 2 it is seen that the best found values of system reliability in all cases are the same with mean values of the same. Again in Case-III and IV the standard deviations of system reliability of the system are zero whereas in Case-I and II, these are interval valued with lower bounds and significantly small widths. Also average CPU time required for implementing the genetic algorithm is also on the lower side.

<table>
<thead>
<tr>
<th>Case</th>
<th>x</th>
<th>Best found value of system reliability ( R )</th>
<th>Mean of ( R )</th>
<th>Standard deviation of ( R )</th>
<th>Mean time-to-failure ( t )</th>
<th>CPU (in sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>(2,5,5,5,5)</td>
<td>[0.999966, 0.999999]</td>
<td>[0.999966, 0.999999]</td>
<td>[0.0000010]</td>
<td>1.0</td>
<td>0.158</td>
</tr>
<tr>
<td>II</td>
<td>(3,5,5,5,5)</td>
<td>[0.999966, 0.999999]</td>
<td>[0.999966, 0.999999]</td>
<td>[0.0000010]</td>
<td>1.0</td>
<td>0.154</td>
</tr>
<tr>
<td>III</td>
<td>(3,4,6,5,4)</td>
<td>[0.999997, 0.999997]</td>
<td>[0.999997, 0.999997]</td>
<td>[0.0]</td>
<td>1.0</td>
<td>0.154</td>
</tr>
<tr>
<td>IV</td>
<td>(2,6,6,5,5)</td>
<td>[0.999998, 0.999998]</td>
<td>[0.999998, 0.999998]</td>
<td>[0.0]</td>
<td>1.0</td>
<td>0.146</td>
</tr>
</tbody>
</table>

7 Conclusions

In this paper, for the first time the reliability optimization problem with Weibull distributed (with interval valued parameters) time-to-failure of each component of a complex system with some resource constraints have been solved. For this purpose, we also developed integer power of an interval valued number. Now, for handling the problem with resource constraints, the corresponding problem has been converted to unconstrained optimization problem with the help of Big-M penalty technique. To solve the problem, we have developed a real coded GA for mixed-integer variables with interval valued fitness function, tournament selection, uniform crossover and one neighborhood mutation for the genes corresponding to integer variables, whole arithmetical crossover and boundary mutation for the gene corresponding to the floating point variable, and elitism of size one. It is well known that the penalty coefficient plays a crucial role in solving constrained optimization problem by penalty function technique.

Therefore, the selection of this parameter is a formidable task. To avoid this difficulty, we have used Big-M penalty technique which does not require any penalty coefficient. In this approach, the value of fitness function is not computed for infeasible solution. Instead, the value of \( M \) is considered for the value of fitness function. However, for infeasible solution, the value of \( M \) may be taken depending on the fitness function value. A small value (in case of maximization problem) or a large value (in case of minimization problem) may be considered for \( M \) to solved constrained optimization problem.
This entire approach opens up scope for reliability optimization when reliability values and Weibull distribution with interval valued parameters are estimated from sample observations and hence vary over interval sets. For further research, one may use the proposed approach in solving real-life application problems relating to the Weibull distribution with interval valued parameters.

Acknowledgement

The first author would like to acknowledge the support of UGC (University Grant Commission), Eastern Regional Office, Kolkata, India, for conducting this research.

References


25. Moore, R.E.: Methods and applications of interval analysis. SIAM, Philadelphia (1979)
