CHAPTER 2

Solution Methodologies

- Interval Approach in Reliability Optimization
- Finite Interval Mathematics
- Interval Order Relations
- Metric Space
- Genetic Algorithm
- GA-Based Constrained Handling Technique
2.1 Interval Approach in Reliability Optimization

During the last few decades, several researchers formulated and solved either single objective or multi-objective reliability optimization problems as integer non-linear programming problems (INLPP) and/or mixed-integer non-linear programming problems (MINLPP) with single or several resource constraints. To solve those problems, they proposed different techniques. In this connection, one may refer to the works of Tillman, Hwang and Kuo (1977a, 1977b and 1980), Nakagawa, Nakashima and Hattori (1978), Misra and Sharma (1991), Chern (1992), Ohtagaki, Nakagawa, Iwasaki, and Narihisa (1995), Kuo, Prasad, Tillman and Hwang (2001), Sun and Li (2002), Gen and Yun (2006), Ha and Kuo (2006b), Coelho (2009a, 2009b) among others. In their works, the design parameters involved in reliability optimization have 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 either new or it 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 are known. In these cases, parameters are said to be imprecise. To tackle the problem with such imprecise parameters, generally stochastic, fuzzy and fuzzy-stochastic approaches are applied and the corresponding problems are converted into deterministic problems for solving them. In the stochastic approach, the parameters are assumed to be random variables with known probability distributions. In the fuzzy approach, the parameters, constraints and goals are considered as fuzzy sets with known membership functions or fuzzy numbers. On the other hand, in the
fuzzy-stochastic approach, some parameters are viewed as fuzzy sets/fuzzy numbers and others as random variables. However, it is a formidable task for a decision-maker to specify the appropriate membership function for fuzzy approach and probability distribution for stochastic approach and both for fuzzy-stochastic approach. So, to avoid these difficulties for handling the imprecise parameters by different approaches, one may use an interval number to represent an imprecise number, as this representation is the most significant representation among others.

### 2.2 Mathematical Backgrounds

#### 2.2.1 Finite Interval Mathematics

An interval number $A$ is a closed interval connected subset of $\mathbb{R}$ denoted by $A = [a_L, a_R]$ and is defined by $A = [a_L, a_R] = \{ x : a_L \leq x \leq a_R, x \in \mathbb{R} \}$, where $a_L$ and $a_R$ are the left and right limits respectively and $\mathbb{R}$ is the set of all real numbers.

An interval $A$ can also be expressed in terms of centre and radius as $A = (c, w) = \{ x : c - w \leq x \leq c + w, x \in \mathbb{R} \}$, where $c$ and $w$ be the centre and radius of the interval $A$ respectively i.e., $a_c = (a_L + a_R)/2$ and $a_w = (a_R - a_L)/2$.

Actually, every real number can be treated as an interval, such as for all $x \in \mathbb{R}$, $x$ can be written as an interval $[x, x]$ having zero width.

Here, we shall give the concise definitions of arithmetical operations like addition, subtraction, multiplication, and division of interval numbers.

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

Then the addition of two intervals $A$ and $B$ is given by

$$A + B = [a_L + b_L, a_R + b_R], \quad A + B = (c + c, w + w)$$

The subtraction of two intervals $A$ and $B$ is given by

$$A - B = [a_L - b_L, a_R - b_R], \quad A - B = (c - c, w - w)$$
\[ A - B = [a_L - b_R, a_R - b_L] \]

or, \[ A - B = A + B = \langle a_c, a_w \rangle - \langle b_c, b_w \rangle = \langle a_c, a_w \rangle + \langle -b_c, b_w \rangle = \langle a_c - b_c, a_w + b_w \rangle. \]

The multiplication of an interval \( A \) by a real number \( \lambda \) is defined by

\[
\lambda A = \begin{cases} 
[\lambda a_L, \lambda a_R] & \text{for } \lambda \geq 0, \\
[\lambda a_R, \lambda a_L] & \text{for } \lambda < 0,
\end{cases}
\]

or, \[ \lambda A = \lambda \langle a_c, a_w \rangle = \langle \lambda a_c, |\lambda| a_w \rangle. \]

The mid-point of an interval \( A \) is denoted by \( m(A) \) and is defined by

\[ m(A) = \frac{a_L + a_R}{2} \]

The product of two different intervals \( A \) and \( B \) is defined by

\[ A \times B = [\min(a_L b_L, a_L b_R, a_R b_L, a_R b_R), \max(a_L b_L, a_L b_R, a_R b_L, a_R b_R)]. \]

The division of the interval \( B \) by the interval \( A \) is defined as

\[ \frac{B}{A} = B \times \frac{1}{A} = [b_L b_R] \times [\frac{1}{a_R}, \frac{1}{a_L}], \text{ provided } 0 \notin [a_L, a_R]. \]

The above definitions are given in the books written by Moore (1979) and Hansen and Walster (2004).

### 2.2.1.1 Integral Power of an Interval

Let \( A = [a_L, a_R] \) be an interval and \( n \) be any non-negative integer number then according to Hansen and Walster (2004) the definition of integer power of an interval is as follows:

\[
A^n = \begin{cases} 
[1, 1] & \text{if } n = 0 \\
[a_L^n, a_R^n] & \text{if } a_L \geq 0 \text{ or if } n \text{ is odd} \\
[a_R^n, a_L^n] & \text{if } a_R \leq 0 \text{ and } n \text{ is even} \\
[0, \max(a_L^n, a_R^n)] & \text{if } a_L \leq 0 \leq a_R \text{ and } n > 0 \text{ is even}
\end{cases}
\]
2.2.1.2 n-th Root of an Interval

According to Karmakar, Mahato and Bhunia (2009), the n-th root of an interval $A = [a_L, a_R]$ is defined as

$$
(A)^{\frac{1}{n}} = \left[ a_L^{\frac{1}{n}}, a_R^{\frac{1}{n}} \right] = \left\{ \begin{array}{ll}
\left[ \sqrt[n]{a_L}, \sqrt[n]{a_R} \right] & \text{if } a_L \geq 0 \text{ or if } n \text{ is odd} \\
[0, \sqrt[n]{a_R}] & \text{if } a_L \leq 0, a_R \geq 0 \text{ and } n \text{ is even} \\
\varnothing & \text{if } a_R < 0 \text{ and } n \text{ is even}
\end{array} \right.
$$

where $\varnothing$ is the empty interval.

2.2.1.3 Rational Power of an Interval

Again applying the definitions of power and different roots of an interval, the rational power of an interval $A = [a_L, a_R]$ is defined as follows:

$$
(A)^{\frac{p}{q}} = (A^p)^{\frac{1}{q}} \text{ or equivalently, } (A)^{\frac{p}{q}} = \exp \left\{ \frac{p}{q} \log A \right\}, \text{ provided } a_L > 0.
$$

2.2.1.4 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 be the same as the real interval arithmetic [Kearfott (1996)].

2.2.1.5 Functions of Finite Interval

Here we shall define different types of functions of interval arguments.

For a monotonically increasing function $f(x)$ in the interval $A = [a_L, a_R]$, where $x \in \mathbb{R}$

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

Using the above definitions the exponential and logarithmic function can be expressed for interval arguments as they are strictly monotonic function.

(i) \( \exp(A) = \exp([a_L, a_R]) = [\exp(a_L), \exp(a_R)] \)

(ii) \( \log(A) = \log([a_L, a_R]) = [\log(a_L), \log(a_R)] \), provided \( a_L > 0 \).

For non-monotonic functions, functions of interval arguments are very much complicated.

For bounded periodic functions

(iii) \( \sin([a_L, a_R]) = [b_L, b_R] \)

where \( b_L = \begin{cases} -1 & \text{if } \exists k \in \mathbb{Z}: 2k\pi - \frac{\pi}{2} \in [a_L, a_R] \\ \min\{\sin(a_L), \sin(a_R)\} & \text{otherwise} \end{cases} \)

and \( b_R = \begin{cases} 1 & \text{if } \exists k \in \mathbb{Z}: 2k\pi + \frac{\pi}{2} \in [a_L, a_R] \\ \max\{\sin(a_L), \sin(a_R)\} & \text{otherwise} \end{cases} \)

(iv) \( \cos([a_L, a_R]) = [b_L, b_R] \)

where \( b_L = \begin{cases} -1 & \text{if } \exists k \in \mathbb{Z}: (2k+1)\pi \in [a_L, a_R] \\ \min\{\cos(a_L), \cos(a_R)\} & \text{otherwise} \end{cases} \)

and \( b_R = \begin{cases} 1 & \text{if } \exists k \in \mathbb{Z}: 2k\pi \in [a_L, a_R] \\ \max\{\cos(a_L), \cos(a_R)\} & \text{otherwise} \end{cases} \)

2.2.1.6 Integration of an Interval Function

According to Moore (1979), the integration of an interval function is defined by

\[
\int_a^b f(y) \, dy = [\int_a^b f_L(y) \, dy, \int_a^b f_R(y) \, dy] \text{ for any } y \in \mathbb{R}.
\]
Here \( f(y) = [f_L(y), f_R(y)] \) and both \( f_L(y) \) and \( f_R(y) \) are continuous real valued functions.

### 2.2.1.7 Interval Power of an Interval

Till now, none has developed the interval power of an interval number. In this thesis, in Chapter 5 and Chapter 8, the interval power of an interval numbers occurs in the formulation of the optimization problems. For this purpose, we have introduced the formula of interval power of an interval as follows:

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

(i) \((A)^B = [a_L, a_R])^{[b_L, b_R]}\)

\[
= \begin{cases} 
\exp(u), \exp(v) & \text{if } a_L \geq 0 \\
a \text{ complex interval} & \text{if } a_L < 0 
\end{cases}
\]

where \( u = \min\{b_L \log a_L, b_L \log a_R, b_R \log a_L, b_R \log a_R\} \)

and \( v = \max\{b_L \log a_L, b_L \log a_R, b_R \log a_L, b_R \log a_R\} \).

(ii) \([-a_L, -a_R])^{[b_L, b_R]}\)

\[
= [a_R, a_L])^{[b_L, b_R]} + \cos[(2k + 1)\pi b_L, (2k + 1)\pi b_R] + i \sin[(2k + 1)\pi b_L, (2k + 1)\pi b_R] \\
\text{if } a_L, a_R \geq 0, k = 0, 1, 2, 3, \ldots
\]

### 2.2.1.8 Mean, Variance and Standard Deviation of Interval Numbers

The mean, variance and standard deviation of \( n \) interval numbers are defined as follows:

Let \( x_i = [x_{iL}, x_{iR}] \), \( i = 1, 2, ..., n \) be the \( i \)-th observation which is an interval number. Then \( \text{mean}(\bar{x}) \), variance \( \text{[Var}(x)\text{]} \) and standard deviation \( (\sigma_x) \) of these numbers are given by
\[ \bar{X} = [\bar{x}_L, \bar{x}_R] = \left[ \frac{1}{n} \sum_{i=1}^{n} x_{iL}, \frac{1}{n} \sum_{i=1}^{n} x_{iR} \right], \]

\[ \text{Var}(x) = [\sigma_L^2, \sigma_L^2] = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{1}{n} \sum_{i=1}^{n} x_{iR} - \frac{1}{n} \sum_{i=1}^{n} x_{iL} \right)^2, \]

and \( \sigma_x = [\sigma_L, \sigma_R] = \sqrt{\text{Var}(x)}. \)

### 2.2.2 Interval Order Relations

For obtaining the optimum solution in solving the optimization problems with interval valued objectives we need to define the order relations of interval numbers.

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

**Type-1**: Two intervals are disjoint [see Figure 2.1].

**Type-2**: Two intervals are partially overlapping [see Figure 2.2].

**Type-3**: One of the intervals contains the other one [see Figure 2.3].

---

**Figure 2.1**: Type-1 intervals

**Figure 2.2**: Type-2 intervals
Studies on Reliability Optimization Problems by Genetic Algorithm

Figure 2.3: Type-3 intervals

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, i.e., $A = B$ iff $a_L = b_L$ and $a_R = b_R$.

In this area, very few researchers defined the order relations of interval valued numbers. Moore (1979) first proposed two order relations of interval numbers.

For any two intervals $A = [a_L, a_R]$ and $B = [b_L, b_R]$, Moore (1979) first gave the two order relations which are as follows:

(i) transitive order relation ‘$<$’ as $A < B$ iff $a_R < b_L$

(ii) transitive order relation set inclusion ‘$\subseteq$’ as $A \subseteq B$ iff $b_L \leq a_L$ and $a_R \leq b_R$.

However, these two order relations cannot order two partially or fully overlapping intervals. Then Ishibuchi and Tanaka (1990) defined the order relations for minimization problems of two closed intervals $A = [a_L, a_R] = \langle a_c, a_w \rangle$ and $B = [b_L, b_R] = \langle b_c, b_w \rangle$ which are as follows:

(i) $A \leq_{LR} B$ iff $a_L \leq b_L$ and $a_R \leq b_R$

$A <_{LR} B$ iff $A \leq_{LR} B$ and $A \neq B$

(ii) $A \leq_{cw} B$ iff $a_c \leq b_c$ and $a_w \leq b_w$

$A <_{cw} B$ iff $A \leq_{cw} B$ and $A \neq B$
These order relations are reflexive, transitive and anti-symmetric i.e., these are partial order. From these definitions it is clear that, for minimization problem, a decision-maker will prefer the interval $A$. Generalizing the definitions of Ishibuchi and Tanaka (1990), Chanas and Kuchta (1996) proposed the concept of $t_{0}, t_{1}$ – cut of an interval for the ranking of interval numbers.

Let $A = [a_{L}, a_{R}]$ be any interval and $t_{0}$ and $t_{1}$ be any two fixed numbers such that $0 \leq t_{0} \leq t_{1} \leq 1$ then the $t_{0}, t_{1}$ – cut of the interval is given by

$$A_{t_{0}, t_{1}} = [a_{L} + t_{0}(a_{R} - a_{L}), a_{L} + t_{1}(a_{R} - a_{L})].$$

According to Chanas and Kuchta (1996), the order relations for the intervals $A = [a_{L}, a_{R}]$ and $B = [b_{L}, b_{R}]$ are as follows:

(i) $A \leq_{LR} l_{[t_{0}, t_{1}]} B$ iff $A_{l_{[t_{0}, t_{1}]}} \leq_{LR} B_{l_{[t_{0}, t_{1}]}}$

$$A <_{LR} l_{[t_{0}, t_{1}]} B$$ iff $A_{l_{[t_{0}, t_{1}]}} <_{LR} B_{l_{[t_{0}, t_{1}]}}$

(ii) $A \leq_{cw} l_{[t_{0}, t_{1}]} B$ iff $A_{l_{[t_{0}, t_{1}]}} \leq_{cw} B_{l_{[t_{0}, t_{1}]}}$

$$A <_{cw} l_{[t_{0}, t_{1}]} B$$ iff $A_{l_{[t_{0}, t_{1}]}} <_{cw} B_{l_{[t_{0}, t_{1}]}}$

After Chanas and Kuchta (1996), Kundu (1997) first noticed that the interval ranking methods discussed earlier could not find the measure ‘How much larger the interval $A$ is, if it is greater than the other?’ He attempted to answer this question by introducing the ‘fuzzy leftness relation’. For the intervals $A$ and $B$, let $a \in A$ and $b \in B$ are uniformly and independently distributed in $A$ and $B$ respectively.

Then $A$ is left to $B$ if $\text{Left}(A, B) = \max\{0, P(a < b) - P(a > b)\} > 0$ and $A$ is right to $B$ if $\text{Right}(A, B) = \max\{0, P(a > b) - P(a < b)\} > 0$, where $P(a < b)$ denotes the probability that $a < b$. This is a probabilistic approach.

In the year 2000, two other approaches of ranking of two intervals were given by Sengupta and Pal (2000). In the first approach, they defined order relations with
respect to the decision-makers’ point of view using the acceptability function
\( \chi : I \times I \rightarrow [0, \infty) \) for the intervals \( A \) and \( B \) as \( \chi(A, B) = \frac{b_c - a_c}{b_w + a_w} \), where \( b_w + a_w \neq 0 \).

\( \chi(A, B) \) may be considered as a grade of acceptability of the ‘first interval to be inferior to the second’. If \( \chi(A, B) = 0 \) then for a minimization problem, the interval \( A \) cannot be accepted as smaller. If \( 0 < \chi(A, B) < 1 \), \( A \) can be accepted with the grade of acceptability \( \frac{b_c - a_c}{b_w + a_w} \). The interval \( A \) is accepted fully if \( \chi(A, B) = 1 \).

According to them, the acceptability index is only a value based ranking index and it can be applied partially to select the best alternative from the pessimistic point of view of the decision-maker. So, only the optimistic decision-maker can use it completely. In another approach, Sengupta and Pal (2000) introduced the fuzzy preference ordering for the ranking of a pair of interval numbers on the real line with respect to a pessimistic decision-makers’ point of view. The fuzzy preference method was described for maximizing the profit interval. However, this method is equally applicable to minimize the cost/time intervals also. In this definition, they assumed that two intervals \( A \) and \( B \) are profit intervals and the problem is to find the maximum profit interval from among them. In this approach, they considered the fuzzy set “Rejection of an interval \( A \) in comparison to the interval \( B \)” or “Acceptance of \( B \) in comparison to \( A \)”.

The membership function of this fuzzy set is given by

\[
\mu(B, A) = \begin{cases} 
1 & \text{if } b_c = a_c \\
\max \left( 0, \frac{b_c - a_L - b_w}{a_c - a_L - b_w} \right) & \text{if } a_L + b_w \leq b_c \leq a_c \\
0 & \text{otherwise}
\end{cases}
\]

This non-linear membership function lies in the interval \([0, 1]\). When the values of
this membership function lies within the interval [0.333, 0.666], this definitions fails to find the order relations.

According to the optimistic and pessimistic decision-makers’ point of view, Mahato and Bhunia (2006) proposed the revised definitions of order relations between interval costs/times for minimization problems and interval profits for maximization problems. Let the two intervals \( A = [a_L, a_R] = [a_u, a_w] \) and \( B = [b_L, b_R] = [b_u, b_w] \) be the uncertain interval costs/time or profits.

Now, we explain their proposed definitions which depend on the decision-makers’ risk taking attitude. In this case, they considered two types of decision-making, viz. (i) Optimistic decision-making and (ii) Pessimistic decision-making.

2.2.2.1 Optimistic Decision-Making

In this decision-making, decision-maker expects the lowest value for minimization problems and highest value for maximization problems ignoring the uncertainty.

**Definition:** For minimization problems, the order relation \( \leq_{\text{omin}} \) between the intervals \( A = [a_L, a_R] \) and \( B = [b_L, b_R] \) is

\[
A \leq_{\text{omin}} B \iff a_L \leq b_L
\]

and \( A <_{\text{omin}} B \iff A \leq_{\text{omin}} B \wedge A \neq B. \)

This implies that \( A \) is superior to (i.e., better than) \( B \) and \( A \) is accepted but \( B \) is not inferior to \( A. \) This order relation is obviously not symmetric.

**Definition:** For maximization problems, the order relation \( \geq_{\text{omax}} \) between the intervals \( A = [a_L, a_R] \) and \( B = [b_L, b_R] \) is

\[
A \geq_{\text{omax}} B \iff a_R \geq b_R
\]

and \( A >_{\text{omax}} B \iff A \geq_{\text{omax}} B \wedge A \neq B. \)
This implies that $A$ is superior to $B$ and the optimistic decision-maker accepts the profit interval $A$. Here also, the order relation $\geq_{\text{omax}}$ is not symmetric.

### 2.2.2.2 Pessimistic Decision-Making

In this decision-making, the decision-maker expects the lowest/highest value with less uncertainty for minimization/maximization problems according to the principle “Less uncertainty is better than more uncertainty”.

**Definition:** For minimization problems, the order relation $<_{\text{pmin}}$ between the intervals $A = [a_L, a_R] = \{a_e, a_w\}$ and $B = [b_L, b_R] = \{b_e, b_w\}$ is

1. $A <_{\text{pmin}} B \iff a_e < b_e$ for Type-1 and Type-2 intervals and
2. $A <_{\text{pmin}} B \iff (a_e \leq b_e) \land (a_w < b_w)$ for Type-3 intervals.

However, for Type-3 intervals with $(a_e < b_e) \land (a_w > b_w)$, a pessimistic decision cannot be taken. Here, the optimistic decision is considered.

**Definition:** For maximization problems the order relation $>_{\text{pmax}}$ between the intervals $A = [a_L, a_R] = \{a_e, a_w\}$ and $B = [b_L, b_R] = \{b_e, b_w\}$ is

1. $A >_{\text{pmax}} B \iff a_e > b_e$, for Type-1 and Type-2 intervals and
2. $A >_{\text{pmax}} B \iff (a_e \geq b_e) \land (a_w < b_w)$, for Type-3 intervals.

Again, for Type-3 intervals with $(a_e > b_e) \land (a_w > b_w)$, a pessimistic decision cannot be taken. In this situation, the optimistic decision may be taken.

### 2.2.2.3 Proposed Definition of Interval Order Relations

In the definitions of Mahato and Bhunia (2006) of pessimistic decision-making of Type-3 intervals, it is observed that sometimes optimistic decisions are to be taken. To overcome this situation, we have proposed two new definitions of order relations.
irrespective of optimistic as well as pessimistic decision-makers’ point of view for maximization and minimization problems separately.

**Definition:** The order relation \( >_{\text{max}} \) between the intervals \( A = [a_L, a_R] = \{a_c, a_w\} \) and \( B = [b_L, b_R] = \{b_c, b_w\} \), then for maximization problems

(i) \( A >_{\text{max}} B \iff a_c > b_c \) for Type-1 and Type-2 intervals and

(ii) \( A >_{\text{max}} B \iff \) either \( a_c \geq b_c \land a_w < b_w \) or \( a_c \geq b_c \land a_R > b_R \) for Type-3 intervals.

According to this definition, the interval \( A \) is accepted for maximization case. Clearly, this order relation \( >_{\text{max}} \) is reflexive and transitive but not symmetric.

**Definition:** The order relation \( <_{\text{min}} \) 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 and

(ii) \( A <_{\text{min}} B \iff \) either \( a_c \leq b_c \land a_w < b_w \) or \( a_c \leq b_c \land a_L < b_L \) 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.

It is to be noted that the definitions given by Mahato and Bhunia (2006) and our proposed definitions give the same results. So these two definitions are equivalent.

### 2.2.3 Metric Space

The term metric is derived from the term metor (measure). The concept of a metric space is essentially due to a French Mathematician M. Frechet, though the definition presently in use is that given by the German Mathematician F. Housdorff in 1914.

**Definition:** Let \( X \) be an arbitrary non-empty set. A mapping \( d : X \times X \to \mathbb{R} \) is said to be metric on \( X \) if it satisfies the following properties:
Studies on Reliability Optimization Problems by Genetic Algorithm

(i) \( d(x, y) \geq 0, \quad \forall x, y \in X \)

(ii) \( d(x, y) = 0 \iff x = y, \quad \forall x, y \in X \)

(iii) \( d(x, y) = d(y, x), \quad \forall x, y \in X \)

(iv) \( d(x, y) \leq d(x, z) + d(z, y), \quad \forall x, y, z \in X \)

If \( X \) is a non-empty set and \( d \) is a metric on \( X \), then \( (X, d) \) is called a metric space. A metric \( d \) is also called a distance function. The real number \( d(x, y) \) is called the distance between \( x \) and \( y \).

Here, we shall give some metric spaces which are used in this thesis.

(i) Let \( X = \mathbb{R}^n \) and suppose \( x = (\xi_1, \xi_2, \ldots, \xi_n) \) and \( y = (\eta_1, \eta_2, \ldots, \eta_n) \) be any two points in \( \mathbb{R}^n \). Define the mapping \( d_p, d_\infty : X \times X \rightarrow \mathbb{R} \) as follows:

\[
d_p(x, y) = \left\{ |\xi_1 - \eta_1|^p + |\xi_2 - \eta_2|^p + \cdots + |\xi_n - \eta_n|^p \right\}^{\frac{1}{p}}, \quad 1 \leq p < \infty
\]

\[
d_\infty(x, y) = \max \left\{ |\xi_1 - \eta_1|, |\xi_2 - \eta_2|, \ldots, |\xi_n - \eta_n| \right\} = \max \left\{ |\xi_i - \eta_i| \right\}
\]

Then, \( d_p \) for each \( 1 \leq p < \infty \) and \( d_\infty \) are metrics on the same underlying set \( X = \mathbb{R}^n \).

(ii) Let \( X = l_p, 1 \leq p < \infty \), be the set of all sequences \( x = (\xi_i) \) of real scalars such that \( \sum_{i=1}^{\infty} |\xi_i|^p < \infty \). Define the mapping \( d : X \times X \rightarrow \mathbb{R} \) by

\[
d(x, y) = \left( \sum_{i=1}^{n} |\xi_i - \eta_i|^p \right)^\frac{1}{p},
\]

where \( x = (\xi_i) \) and \( y = (\eta_i) \) are in \( l_p \).

2.3 Solution Methodologies

2.3.1 Genetic Algorithm

Genetic algorithm (GA) is a well-known stochastic search iterative method based on the evolutionary theory of Charles Darwin “survival of the fittest” and natural
Solution methodologies

Genetics. GA has successfully been applied to optimization problems in different fields, like engineering design, reliability optimization, optimal control, transportation and assignment problems, job scheduling, inventory control and other real-life decision-making problems. The most fundamental idea of Genetic Algorithm is to imitate the natural evolution process artificially in which populations undergo continuous changes through genetic operators, like crossover, mutation and selection. The concept of GA was first introduced by Prof. John Holland of the University of Michigan, Ann Arbor. He is considered to be the father of GA. His idea of genetic algorithm was first used to solve optimization problem by De-Jang (1975). Thereafter, a researcher has contributed much to the major development of this field. Most of the initial research work in this field can be found in several International Conference Proceedings. The detailed discussion of genetic algorithms, including extensions along with related topics, can be found in the books on GA [Holland (1975), Goldberg (1989), Davis (1991), Michalewicz (1996), Mitchell (1996), Gen and Cheng (1997) and Vose (1999)].

Genetic algorithm can easily be implemented with the help of computer programming. In particular, it is very useful for solving complicated optimization problems which cannot be solved easily by direct or gradient based mathematical techniques. It is very effective to handle large-scale, real-life, discrete and continuous optimization problems without making unrealistic assumptions and approximations. Keeping the imitation of natural evolution as the foundation, genetic algorithm can be designed appropriately and modified to exploit special features of the problem to solve. This algorithm starts with an initial population of probable solutions, called individuals, to a given problem where each individual is represented using different form of coding as a chromosome. These chromosomes are evaluated for their fitness.
Based on their fitness, chromosomes in the population are to be selected for reproduction and selected individuals are manipulated by two known genetic operations, like crossover and mutation. The crossover operation is applied to create offspring from a pair of selected chromosomes. The mutation operation is used for a slight modification/change to reproduce offspring. The repeated applications of genetic operators to the relatively fit chromosomes result in an increase in the average fitness of the population over generation and identification of improved solutions to the problem under investigation. This process is applied iteratively until the termination criterion is satisfied.

2.3.1.1 GA Terminology

It is important to first understand the terminology that is used with respect to genetic algorithm. Some of the commonly used terms are as follows:

- **Population**: A collection of several alternate solutions to the given problem
- **Population size**: The population size determines the amount of information stored by the GA.
- **Chromosome**: Each individual in the population is called a chromosome.
- **Genes**: Often these individuals are coded as binary/real strings and the individual character or symbol in the string is named as genes.
**Fitness function**: It is an evolution function, which is used to determine the fitness of each chromosome. The fitness function is usually user defined and problem specific.

**Solution space**: The range of possible solutions is referred to as the solution space and the cost and the fitness of each point is referred to as the altitude in the landscape of the problem.

**Generation gap**: It is the fraction of the individuals in the population that are replaced from one generation to the next and is equal to one for simple GA.

**Termination criterion**: The termination criterion 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 criterion.

(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.

(iii) The number of generations reaches a prescribed finite number of generation (called maximum number of generations).
The procedural algorithm of the working principle of GA is as follows:

2.3.1.2 Algorithm

Step-1 : Set population size \((p\text{\_size})\), crossover probability \((p\text{\_cross})\), mutation probability \((p\text{\_mute})\), maximum generation \((\text{max\_gen})\) and bounds of the variables.

Step-2 : \(t = 0 \) \([t \text{ represents the number of current generation}]\).

Step-3 : Initialize the chromosome of the population \(P(t)\) \([P(t) \text{ represents the population at } t\text{-th generation}]\).

Step-4 : Evaluate the fitness function of each chromosome of \(P(t)\) considering the objective function as the fitness function.

Step-5 : Find the best chromosome from the population \(P(t)\).

Step-6 : \(t\) is increased by unity.

Step-7 : If the termination criterion is satisfied go to Step-14, otherwise, go to next step.

Step-8 : Select the population \(P(t)\) from the population \(P(t-1)\) of earlier generation by tournament selection process.

Step-9 : Alter the population \(P(t)\) by crossover, mutation and elitism operators.

Step-10 : Evaluate the fitness function value of each chromosome of \(P(t)\).

Step-11 : Find the best chromosome from \(P(t)\).

Step-12 : Compare the best chromosome of \(P(t)\) and \(P(t-1)\) and store better one.


Step-14 : Print the best chromosome \(\text{which is the solution of the optimization problem}\).

Step-15 : End.
To implement the GA, the following basic components are to be considered:

(i) GA parameters (population size, maximum number of generation, crossover rate and mutation rate)

(ii) Chromosome representation

(iii) Initialization of population

(iv) Evaluation of fitness function

(v) Selection process

(vi) Genetic operators (crossover, mutation and elitism)

2.3.1.3 GA Parameters

There are several GA parameters, viz. population size \((p\_size)\), maximum number of generation \((max\_gen)\), crossover rate i.e., the probability of crossover \((p\_cross)\) and mutation rate i.e., the probability of mutation \((p\_mute)\). There is no hard and fast rule for selecting the population size for GA, how large it should be. The population size is problem dependent and will need to increase with the dimensions of the problem. If the population size is very large, storing of data in intermediate steps of GA may arise some difficulties at the time of execution. When the population size is very small, some genetic operators do not work properly. However, population size is restricted by both time complexity and space complexity. Regarding the maximum number of generations, there is no clear indication for considering this value. It varies from problem to problem and depends upon the number of genes (variables) of a chromosome and prescribed as stopping/termination criteria to make sure that the solution has converged. From natural genetics, it is obvious that the rate of crossover is always greater than that of rate of mutation. Generally, the crossover rate varies from 0.60 to 0.95 whereas the mutation rate varies from 0.05 to 0.20. Sometimes
mutation rate is considered as \( l/n \) where \( n \) is the number of genes (variables) of the chromosome.

### 2.3.1.4 Representation of Chromosomes

To represent an appropriate chromosome is an important issue in the application of GA for solving the optimization problem and users of GA face a hard situation how to represent the appropriate chromosome (individual). There are different types of representations, like, binary, real, octal, hexadecimal coding, available in the existing literature. Among different representations, mainly binary coding and real coding representations are very popular. In the initial implementation of GAs, the chromosomes were represented by the strings of binary numbers. In this representation, binary substrings of each variable with the desired precision are concatenated to represent an individual. As a result, the string length of an individual will be large. In this case, genetic algorithm would perform poorly. To overcome these difficulties, real numbers are used to represent the chromosomes in GAs. In this case, a chromosome is coded in the form of vector/matrix of integer/ floating point or combination of the both numbers and every component of that chromosome represents a decision variable of the problem. In this representation, each chromosome is encoded as a vector of integer/ floating or combination of the both numbers, with the same component as the vector of decision variables of the problem. This type of representation is accurate and more efficient as it is closed to the real design space and moreover, the string length of each chromosome is the number of design variables. In this representation, for a given problem with \( n \) decision variables, a \( n \)-component vector \( x = (x_1, x_2, \ldots, x_n) \) is used as a chromosome to represent a solution to the problem. A chromosome denoted as \( v_k \) (\( k = 1, 2, \ldots, p_{\text{size}} \)) is an ordered list of \( n \) genes as \( v_k = \{v_{k1}, v_{k2}, \ldots, v_{kn} \} \).
2.3.1.5 Initialization

After representation of chromosome, the next step is to initialize the chromosome that will take part in the artificial genetics. To initialize the population, first of all we have to find the independent variables and their bounds for the given problem. Then the initialization process produces population size number of chromosomes in which every component for each chromosome is randomly generated within the bounds of the corresponding decision variable. There are several procedures for selecting a random number of either integer type or float point type. In our whole work, we have used the following algorithm for selecting of an integer random number.

An integer random number between \( a \) and \( b \) can be generated as either \( x = a + g \) or \( x = b - g \)

where \( g \) is a random integer between 1 and \( |a - b| \).

On the other hand, to generate a floating point random number, we have used uniform distribution as follows:

**Uniform Distribution**

Using this distribution, a random number on an interval \([a, b]\) can be generated as \( x = a + r(b - a) \) where \( r \) is another random number between 0 and 1.

The initialization procedure produces \( p\_size \) chromosomes where \( p\_size \) denotes the population size, by the following algorithm.

**Step-1:** Generate a random number between 0 and 1.

**Step-2:** Assign \( x_i = l_i + r(u_i - l_i) \), where \( l_i \) and \( u_i \) are lower and upper bounds of \( x_i \).

**Step-3:** Repeat the steps 1 and 2 for \( n \) times and produce a vector \( \{x_1, x_2, ..., x_n\} \).

**Step-4:** Repeat the steps 1, 2 and 3 for \( p\_size \) times and produce \( p\_size \) initial feasible solutions.
Step-5: Stop.

It may be noted that a random value can be selected alternatively from the discrete set of values \{0, 0.1, 0.2, 0.3, ..., 0.9\}. Generally, this process is used to find the low precision solutions of a problem. However, if a solution has a high-precision value, a random number \( r \) is selected from either discrete set of values or Step-1 of earlier algorithm. Again, for getting the boundary points in a chromosome by uniform distribution, random value selection from discrete values is very efficient.

2.3.1.6 Evaluation/ Fitness Value Computation

Evaluation/fitness function plays an important role in GA. This role is same for natural evolution process in the biological and physical environments. After initialization of chromosomes of potential solutions, we need to see how relatively good they are. Therefore, we have to calculate the fitness value for each chromosome. In our work, the value of objective function of the reduced unconstrained optimization problems corresponding to the chromosome is considered as the fitness value of that chromosome.

2.3.1.7 Selection

The selection operator which is the first operator in artificial genetics plays an interesting role in GA. This selection process is based on the Darwin’s principle on natural evolution “survival of the fittest”. The primary objective of this process is to select the above average individuals/chromosomes from the population according to the fitness value of each chromosome and eliminate the rest of the individuals/chromosomes. There are several methods for implementing the selection process.

Some of the well known selection operators are as follows:
(a) Ranking Selection

(b) Roulette wheel selection

(c) Tournament selection

(d) Stochastic Universal Sampling selection

(e) Steady state selection

In our whole work, we have solved different types of constrained optimization problems. As a result, for solving those problems we have used only the tournament selection with size two. In this selection, two individuals in the population are selected based on their fitness. The following assumptions for this selection procedure are to be considered:

(i) *when both the individuals/chromosomes are feasible then the one with better fitness value is selected.*

(ii) *when one individual/chromosome is feasible and another is infeasible then the feasible one is selected.*

(iii) *when both the individuals/chromosomes are infeasible with unequal constraint violation, then the chromosome with less constraint violation is selected.*

(iv) *when both the individuals/chromosomes are infeasible with equal constraint violation, then any one individual/chromosome is selected.*

2.3.1.8 Crossover

The exploration and exploitation of the solution space can be made possible by exchanging genetic information of the current chromosomes. After the selection process, other genetic operators, like crossover and mutation are applied to the resulting chromosomes those which have survived. Crossover is an operator that creates new individuals/chromosomes (offspring) by combining the features of both parent solutions. It operates on two or more parent solutions at a time and produces
offspring for next generation. In this operation, expected \([p_{cross} \times p_{size}]\) number of chromosomes will take part (* and [ ] denote the product and the integral value respectively).

In this thesis, the following crossover operators have been used:

(a) Whole arithmetical crossover (for floating point variables)

(b) Intermediate crossover (for integer variables)

The different steps of general crossover operation are as follows:

**Step-1:** Find the integral value of \([p_{cross} \times p_{size}]\) and store it in \(N\).

**Step-2:** Select two parent chromosomes \(v_k\) and \(v_i\) randomly from the population.

**Step-3:** Compute the components \(v_{kj}\) and \(v_{ij}\) (\(j = 1, 2, \ldots, n\)) of two offspring from the parent chromosomes \(v_k\) and \(v_i\).

**Step-4:** Repeat Step-2 and Step-3 for \(\frac{N}{2}\) times.

In case of whole arithmetic crossover, components \(v_{kj}\) and \(v_{ij}\) (\(j = 1, 2, \ldots, n\)) of two offspring will be created by

\[
\bar{v}_{kj} = c v_{kj} + (1-c)v_{ij} \\
\bar{v}_{ij} = (1-c)v_{kj} + c v_{ij}
\]

where \(c\) is a random number between 0 and 1.

In case of intermediate crossover, components \(v_{kj}\) and \(v_{ij}\) (\(j = 1, 2, \ldots, n\)) of two offspring will be created by

\[
\bar{v}_{kj} = v_{kj} - g \text{ and } \bar{v}_{ij} = v_{ij} + g \text{ if } v_{kj} > v_{ij} \\
\text{or, } \bar{v}_{kj} = v_{kj} + g \text{ and } \bar{v}_{ij} = v_{ij} - g
\]

where \(g\) is a random integer number between 0 and \(|v_{kj} - v_{ij}|\), \(j = 1, 2, \ldots, n\).
The aim of mutation operator is to introduce the random variations into the population and is used to prevent the search process from converging to the local optima. This operator helps to regain the information lost in earlier generations and is responsible for fine tuning capabilities of the system and is applied to a single individual only. Usually, its rate is very low; because otherwise it would defeat the order building being generated through the selection and crossover operations.

The different steps of mutations operations are as follows:

**Step-1:** Find the integral value of $\left\lfloor p_{\text{mute}} \times \text{p\_size} \right\rfloor$ and store it in $N$.

**Step-2:** Select a chromosome $v_i$ randomly from the population.

**Step-3:** Select a particular gene $v_{ik} (k = 1, 2, \ldots, n)$ on chromosome $v_i$ for mutation and domain of $v_{ik}$ is $[l_{ik}, u_{ik}]$.

**Step-4:** Create new gene $v'_{ik} (k = 1, 2, \ldots, n)$ corresponding to the selected gene $v_{ik}$ by mutation process.

**Step-5:** Repeat Step-2 to Step-4 for $N$ times.

In this thesis, the following mutation operators have been used:

(a) Uniform mutation (for integer variables)

(b) One-neighborhood mutation (for integer variables)

(c) Boundary mutation (floating point variables)

Among these mutation operators, one-neighborhood mutation is new. For the first time, we have proposed this operator. Basically, this operator is used in GA to mutate the gene corresponding to integer variables. Other two operators are well known mutation operators available in the existing literature.

After mutation process, let $v'_{ik} (k = 1, 2, \ldots, n)$ be the mutated gene corresponding to the selected gene $v_{ik}$.
In case of one-neighborhood mutation,

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

In case of boundary mutation,

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

In case of uniform mutation

\[
\begin{align*}
\v_{ik}^{'} &= \begin{cases} 
     v_{ik} + \Delta(u_{ik} - v_{ik}) & \text{if a random digit is 0} \\
     v_{ik} - \Delta(v_{ik} - l_{ik}) & \text{if a random digit is 1}
\end{cases}
\end{align*}
\]

where \(\Delta(y)\) returns a value in the range \([0, y]\).

2.3.1.9 Elitism

In any generation of GA, sometimes there arises a situation when the best chromosome may get lost from the population when a new population is created by crossover and mutation operations. To overcome this situation the worst individual/chromosome of the current generation is replaced by the best individual/chromosome of previous generation. Instead of single chromosome one or more chromosomes may take part in this operation. This process is named as elitism.

2.3.1.10 Advantages and Disadvantages of Genetic Algorithm

The main advantages of Genetic algorithm are as follows:

(i) can easily be implemented.
(ii) optimizes the objective function with continuous, discrete, permutation and mixed variables.

(iii) does not normally require derivative information.

(iv) deals with large number of decision/design variables.

(v) produces global optimum, does not stuck to local optimum.

(vi) is problem, as well as variable independent.

(vii) gives alternative solution (near optimum).

(viii) works not only with the analytical functions, but also works with experimental data.

(ix) works with a set of solutions instead of single solution in each iteration/generation.

(x) is able to solve problem with non-convex solution space, where classical methods usually fail.

(xi) It also performs well with problems where the objective function is interval valued and highly non-linear, discontinuous or has many local optima.

All these advantages make the GAs superior from the classical optimization techniques in real world applications, mainly for very complicated engineering/scientific problems. Though there are several advantages of GA in solving different types of optimization problems, there are few disadvantages also. These are as follows:

(i) It is often seen that a genetic algorithm caught in a local optimum, and that all or most of the population concentrates on a small part of the search space located around the local optimum. This is usually known as premature convergence.

(ii) The most difficult and time consuming issue in the successful application of GAs is to determine the approximate settings of GA parameters. The parameters of
genetic algorithm need to be tuned for efficiency. However, Michalewicz (1996) mentioned that the determination of proper values of these genetic parameters is an art and the quality of this tuning gently depends on the user-experience as well as their knowledge of the problem.

(iii) Computational efficiency can be lower than that of other methods.

2.3.2 GA-Based Constrained Handling Technique

In applications of GA for solving 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 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. 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. When the objective function is increased/decreased with a penalty term multiplied by so called penalty coefficient there arises a difficulty to select the initial value and upgrading strategy for the penalty coefficient. To overcome this difficulty, Deb (2000) proposed a GA-based Parameter Free Penalty (PFP) technique. In this technique, the worst fitness value of GA for feasible solutions is considered as the fitness value of infeasible solution without multiplying the penalty coefficient i.e., the fitness function values of infeasible solutions are independent of the objective function value for the same solution.

Let us consider a constrained optimization problem with interval objective
Maximize \([f_L(x), f_R(x)]\) \hspace{1cm} (2.1)

subject to the constraints

\[g_j(x) \leq b_j, \quad j = 1, 2, \ldots, m,\]

Therefore, according to the PFP technique, the converted problem of problem (2.1) is as follows:

Maximize \([\hat{f}_L(x), \hat{f}_R(x)]\)

\[= [f_L(x), f_R(x)] - \left[ \sum_{j=1}^{m} \max(0, g_j(x) - b_j), \sum_{j=1}^{m} \max(0, g_j(x) - b_j) \right] + \theta(x) \] \hspace{1cm} (2.2)

where \[\theta(x) = \begin{cases} [0, 0] & \text{if } x \in S \\ [-f_L(x), f_R(x)] + \min[f_L(x), f_R(x)] & \text{if } x \notin S \end{cases}\]

and \(S = \{ x : g_j(x) \leq b_j, j = 1, 2, \ldots, m \}\)

Here the parameter \(\min[f_L(x), f_R(x)]\) is the value of interval valued objective function of the worst feasible solution in the population. Alternatively, the problem may be solved with another fitness function by penalizing a large positive number \((\text{say } M \text{ which can be written in the interval form as } [M, M])\) [Gupta, Bhunia and Roy (2009)]. We denote this penalty as Big-M penalty and its form is as follows:

Maximize \([\hat{f}_L(x), \hat{f}_R(x)] = [f_L(x), f_R(x)] + \theta(x)\) \hspace{1cm} (2.3)

where \[\theta(x) = \begin{cases} [0, 0] & \text{if } x \in S \\ [-f_L(x), f_R(x)] + [-M, -M] & \text{if } x \notin S \end{cases}\]

and \(S = \{ x : g_j(x) \leq c_j, j = 1, 2, \ldots, m \text{ and } l \leq x \leq u \}\)

The above problems (2.2) and (2.3) are non-linear unconstrained optimization problem with interval valued objective.

In our work, we have used Big-M penalty technique with the value of \(M\) as 99999.