Differential evolution (DE) is an optimization algorithm developed by Price and Storn, which solves real valued problems based on the principles of natural evolution \[110,111\]. DE uses a pair of vector population both of which contain \(N_p\) \(D\)-dimensional vectors of real valued parameter. The current population symbolized by \(P^{(G)}\) is composed of floating point encoded individuals \(X_i\). These vectors contain many parameters as the problem decision variables \(D\). The population size \(N_p\) is an algorithm control parameter selected by the user that remains constant throughout the optimization process.

\[
P^{(G)} = [X^{(G)}_1, \ldots, X^{(G)}_{N_p}]
\]

\[
X^{(G)}_i = [X^{(G)}_{1,i}, \ldots, X^{(G)}_{D,i}]^T, \quad i = 1, \ldots, N_p
\]

The optimization process in differential evolution contains various operations like mutation, crossover and selection.
2.1 Initialization

Before initializing the population, both maximum (upper) and minimum (lower) bounds for each parameter must be specified. Once the initialization bounds have been specified, a random number generator $\text{rand}_j(0,1)$ generates a uniformly distributed random number from within the range $[0,1]$.

\[
X_{j,i}^{(0)} = X_{j}^{\text{min}} + \text{rand}_j(0,1).(X_{j}^{\text{max}} - X_{j}^{\text{min}})
\]  

(2.2)

where $i = 1,...,N_p$ and $j = 1,...,D$; $X_{j}^{\text{min}}$ and $X_{j}^{\text{max}}$ are the lower and upper bounds of the $j^{th}$ decision parameter. The subscript $j$ indicates that a new random value is generated for each decision variable.

2.2 Mutation

Once initialized, DE mutates and recombines the population to produce a population of $N_p$ trial vectors. In particular, differential mutation adds a scaled, randomly sampled, vector difference to a third vector as shown in Equation 2.3 where $(X'_i)$ is the mutant vector.

\[
X'_i^{(G)} = X_a^{(G)} + F(X_b^{(G)} - X_c^{(G)}), \quad i = 1,...,N_p
\]

(2.3)

The base vector index, $a$ is assumed to be a randomly chosen vector index that is different from the target vector index, $i$. Except for being distinct from each other and from both the base and target vector indices, the difference vector indices $b$ and $c$, are also randomly selected once per mutant and hence $a \neq b \neq c \neq i$. The scaling constant $(F) \in (0, 1+)$ is a real number that controls the rate at which the population evolves. Scaling vector differences ensure that trial vectors do not duplicate the target vector and
thus the search space is explored more thoroughly avoiding the premature convergence. Figure 2.1 illustrates the construction of mutant vector \( (X'_i) \) in a two-dimensional parameter space.

\[
X'_i = X^{(G)}_i + F(X^{(G)}_b - X^{(G)}_c)
\]

Figure 2.1: Differential mutation

### 2.3 Crossover

DE employs uniform crossover to build/generate trial vectors \( (X''_i) \), by mixing the mutant vectors and target vectors \( X_i \) according to a selected probability distribution

\[
X''_{j,i}^{(G)} = \begin{cases} 
  X'^{(G)}_{j,i} & \text{if } \text{rand}_j(0, 1) \leq C_r \text{ or } j = j_{\text{rand}} \\
  X^{(G)}_{j,i} & \text{otherwise}
\end{cases} \tag{2.4}
\]

The crossover probability, \( (C_r) \in (0, 1) \) is a user defined value that controls the fraction of parameter values that are copied from the mutant. To determine the source of a given parameter, a uniform crossover is used to compare the
output of a uniform random number generator, \( rand_j(0,1) \). It is presumed that if the random number \( \leq (C_r) \), the trial parameter is inherited from the mutant \( (X_i^G) \), otherwise the parameter is copied from vector \( X_i^G \). In addition, the trial parameter with randomly chosen index \( j_{rand} \), is taken from the mutant to ensure that the trial does not duplicate, \( X_i^G \). Because of this additional demand, \( (C_r) \) only approximates the true probability \( p_{cr} \), that a trial parameter will be inherited from the mutant. Figure 2.2 plots the possible trial vectors that can result from uniformly crossing a mutant vector \( (X_i^G) \), with the vector, \( X_i^G \).

![Figure 2.2: Uniform crossover](image)

2.4 Selection

Finally, the selection operator determines the population by choosing between the trial vectors and their predecessors (target vectors). If the trial vector, \( (X_i''^G) \) has an equal or lower fitness function value (optimal) than that of its target vector \( X_i^G \), it replaces the target vector in the next generation.
Otherwise, the target vector retains its place in the population for at least one more generation as given by equation 2.5.

\[
X_i^{(G+1)} = \begin{cases} 
    X_i''^{(G)} & \text{if } f(X_i''^{(G)}) \leq f(X_i^{(G)}), \ i = 1, \ldots, N_p \\
    X_i^{(G)} & \text{otherwise}
\end{cases}
\] (2.5)

By comparing each trial vector with the target vector from which it inherits parameters, DE more tightly integrates crossover and selection than do other evolutionary algorithms. Once the new population is installed, the process of mutation, crossover and selection is repeated for several generations, allowing the individuals to improve their fitness as they explore the solution space in search for optimal values.

### 2.5 Various strategies of DE

DE offers several variants or strategies for optimization. These can be denoted by $DE/x/y/z$, where $x$ refers to the vector used to generate mutant vectors, $y$ is the number of difference vectors used in the mutation process and $z$ is the crossover scheme used in the crossover operation (i.e., binomial and exponential).

#### 2.5.1 Binomial crossover (or) uniform crossover

Uniform crossover is a process in which independent random trials determine the source for each trial parameter. Crossover is uniform in the sense that each parameter, regardless of its location in the trial vector, has the same probability, $p_{cr}$ of inheriting its value from a given vector. DE’s version of uniform crossover begins by taking a randomly chosen parameter from the mutant so that the trial vector will not replicate the target vector. Comparing
$C_r$ to $\text{rand}_j(0, 1)$ determines the source for each remaining trial parameter. If $\text{rand}_j(0, 1) \leq C_r$, then the parameter comes from the mutant, otherwise, the target is the source.

### 2.5.2 Exponential crossover

DE’s exponential crossover works by choosing one parameter randomly and copying from the mutant to the corresponding trial vector so that the trial vector will be different from the vector with which it will be compared. The source of subsequent trial parameters is determined by comparing $C_r$ to a uniformly distributed random number $\text{rand}_j(0, 1)$ that is generated anew for each parameter. As long as $\text{rand}_j(0, 1) \leq C_r$, parameters continue to be taken from the mutant vector, but the first time that $\text{rand}_j(0, 1) > C_r$, the current and all remaining parameters are taken from the target vector.

Storn and Price [111] have proposed ten different working strategies for DE as listed below:

1. $DE/best/1/exp$
2. $DE/rand/1/exp$
3. $DE/best/2/exp$
4. $DE/rand/2/exp$
5. $DE/best/1/bin$
6. $DE/rand/1/bin$
7. $DE/rand/2/bin$
8. $DE/best/2/bin$
9. $DE/rand – to – best/1/exp$
Many studies have been carried out to compare the effectiveness of these strategies on several bench mark problems. It is found that the sixth strategy of DE i.e., \( DE/rand/1/bin \) works well than the other strategies [112–115]. Hence this strategy is adapted for the case studies taken up in the present work. A detailed algorithm of DE is given in the flow chart as shown in the figure 2.3.

### 2.6 Advantages and applications of DE

The advantages of DE over other optimization techniques can be listed as below. DE –

- is very simple and easy to implement
- is efficient in utilizing memory
- poses lower computational complexity and hence scales better on large problems
- takes lesser computational effort and hence converges fast
- offers greater freedom in designing a mutation distribution
- encodes all parameters as floating point numbers regardless of type (integer or discrete)

DE has been widely used in various science and engineering fields such as electromagnetics [116], power system optimization [117], Chaotic systems [118], engineering design problems [119, 120] etc.
Chapter 2. Differential evolution

Start

Read the parameters - Scaling factor $F$, Crossover constant $C_r$,
Population size $N_P$, Maximum iterations $G_{max}$ and Decision variables $D$

Set the iteration counter $G = 0$

Set the population index $i = 1$

Set the decision variable $j = 1$

Initialize the parent vectors uniformly in the random search space

$X_{i,j}^{(0)} = X_{min}^{j} + \text{rand}(0,1) \cdot (X_{max}^{j} - X_{min}^{j})$

Is $j > D$?

Is $i > N_P$?

No

Yes

$j = j + 1$

Is $i > N_P$?

No

Yes

$i = i + 1$

Find the fittest parent vector among entire population & set it as the target vector, $t_v$

Is $G > G_{max}$?

Yes

Print the results

No

Set the iteration counter $G = G + 1$

Set the decision variable $j = 1$

Set the population index $i = 1$

Perform mutation

$X_i^{(G)} = X_a^{(G)} + F(X_b^{(G)} - X_c^{(G)})$

Is $i > N_P$?

No

$j = j + 1$

Yes

Is $j > D$?

No

Yes

$i = i + 1$

Perform crossover

$X_{j,j}^{(G)} = \begin{cases} 
X_{i,j}^{(G)}, \text{ if } \text{rand}j(0,1) \leq C_r \\
X_{i,j}^{(G)}, \text{ otherwise}
\end{cases}$

Perform selection

$X_i^{(G+1)} = \begin{cases} 
X_i^{(G)}, \text{ if } f(X_{i,j}^{(G)}) \leq f(X_i^{(G)}) \\
X_i^{(G)}, \text{ otherwise}
\end{cases}$

Figure 2.3: Flow chart of the differential evolution algorithm