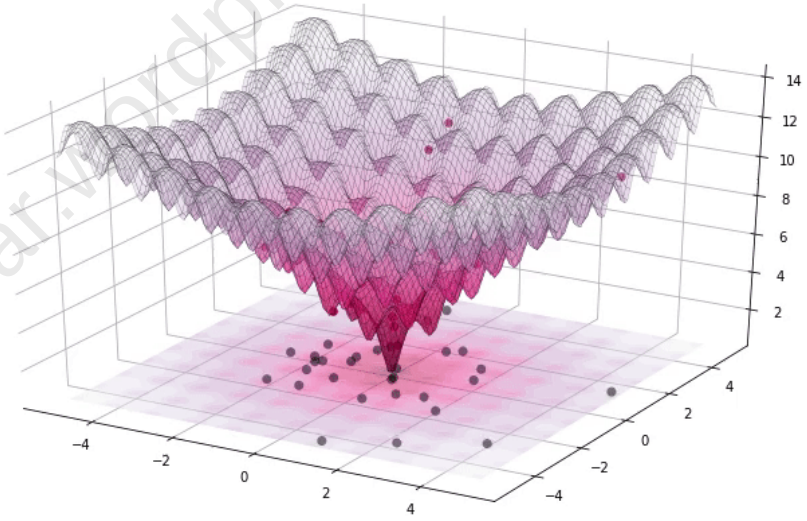
Differential Evolution and Variants

Rajesh Kumar, PhD, PDF (NUS, Singapore)

SMIEEE (USA), FIET (UK) FIETE, FIE (I), SMIACSIT, LMISTE, MIAENG
Professor , Department of Electrical Engineering
Malaviya National Institute of Technology, Jaipur, India,
Mobile: (91) 9549654481
Email: rkumar.ee@mnit.ac.in, rkumar.ee@gmail.com
Web: http://drrajeshkumar.wordpress.com/

What is Differential Evolution(DE)?

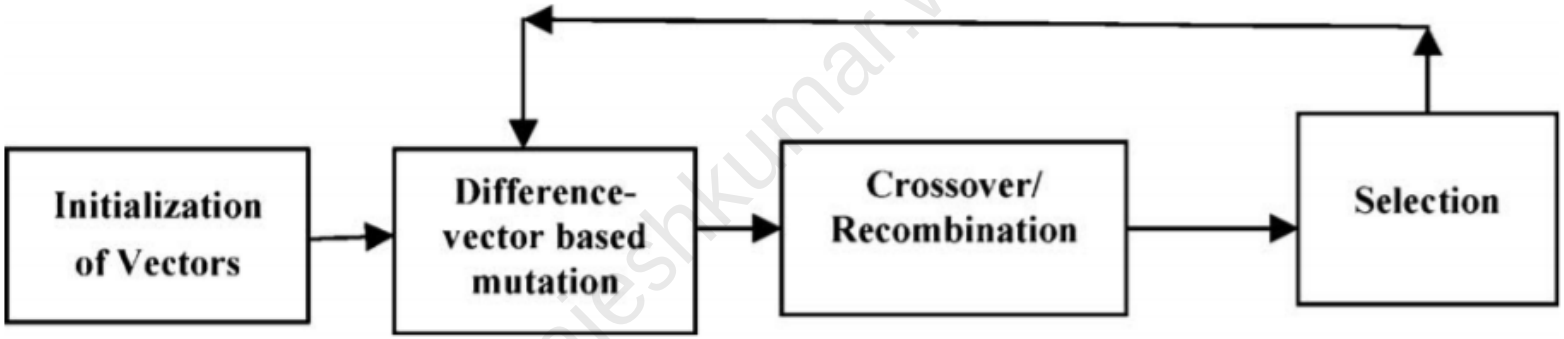
- DE is a population-based optimization algorithm.
- The algorithm was introduced by Storn and Price in 1996.
- Developed to optimize real parameter, real valued functions.



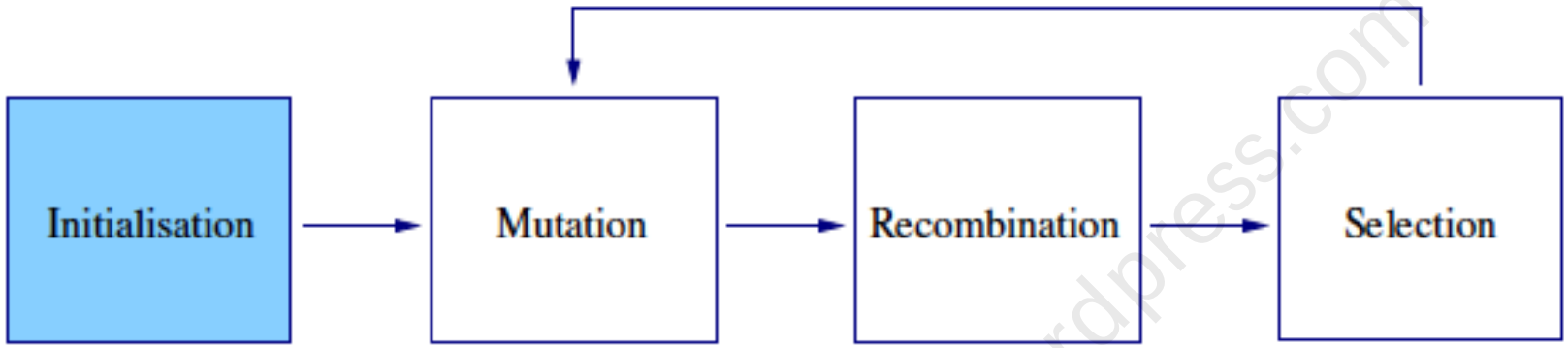
Differential Evolution

- DE is a stochastic, population-based optimization algorithm for solving nonlinear optimization problem.
- The population is composed of N_p individuals
- Every individual in population represents a possible solution
- DE operates in three consecutive steps in every iteration
  - Mutation
  - Crossover
  - Selection

Main stages of DE algorithm



Initialize the population

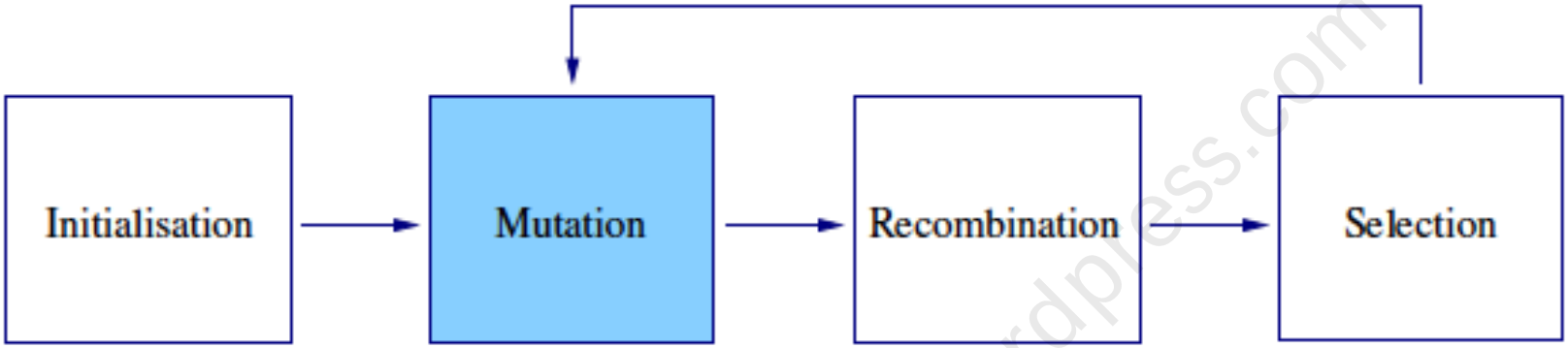


- Define upper and lower bounds for each parameter:

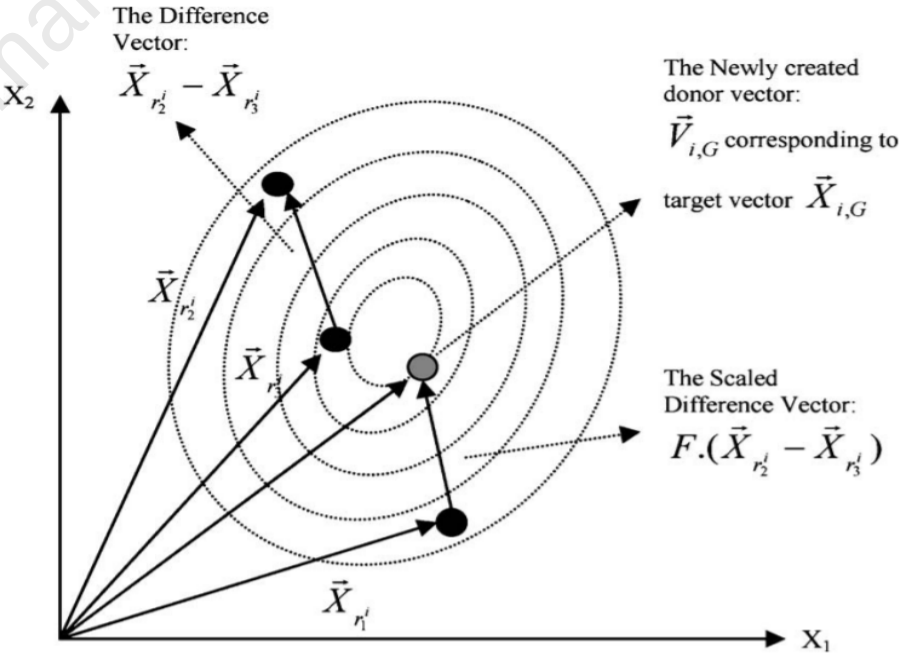
$$x_j^L \leq x_{j,i,1} \leq x_j^U$$

- Randomly select the initial parameter values uniformly on the intervals $[x_j^L, x_j^U]$

Mutation



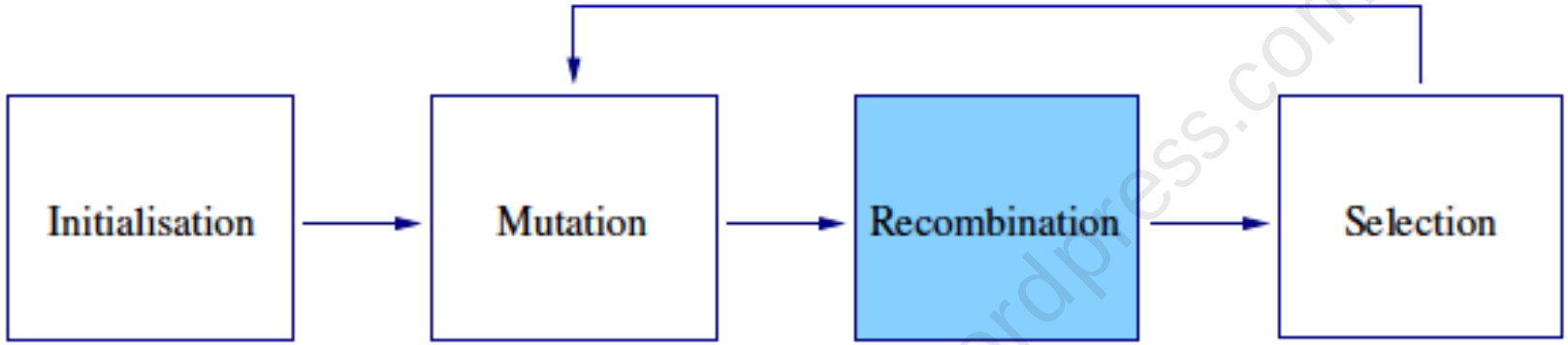
- A parent vector from the current generation is called target vector.
- A mutant vector obtained through the differential mutation operation is known as donor vector.
- Finally an offspring formed by recombining the donor with the target vector is called trial vector.



Different Mutation Strategies

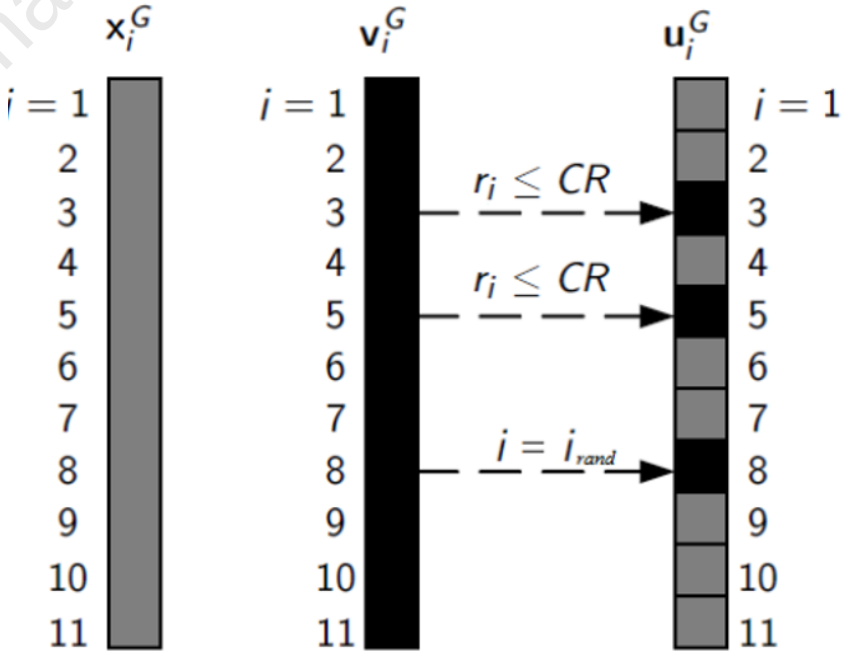
Strategy	Formulation
1. DE/best/1/exp	$V_{(G+1)} = V_{(best,G)} + F\left[V_{(1,G)} - V_{(2,G)}\right]$
2. DE/rand-to-best/1/exp	$V_{(G+1)} = V_{(i,G)} + \lambda \cdot \left[V_{(1,G)} - V_{(2,G)}\right] + F \cdot \left[V_{(1,G)} - V_{(2,G)}\right]$
3. DE/best/2/exp	$V_{(G+1)} = V_{(best,G)} + F\left[V_{(1,G)} + V_{(2,G)} - V_{(3,G)} - V_{(4,G)}\right]$
4. DE/rand/2/exp	$V_{(G+1)} = V_{(5,G)} + F\left[V_{(1,G)} + V_{(2,G)} - V_{(3,G)} - V_{(4,G)}\right]$
5. DE/best/1/bin	$V_{(G+1)} = V_{(best,G)} + F \cdot \left[V_{(2,G)} - V_{(3,G)}\right]$
6. DE/rand/1/bin	$V_{(G+1)} = V_{(1,G)} + F \cdot \left[V_{(2,G)} - V_{(3,G)}\right]$
7. DE/rand-to-best/1/bin	$V_{(G+1)} = V_{(i,G)} + \lambda \cdot \left[V_{(best,G)} - V_{(i,G)}\right] + F \cdot \left[V_{(1,G)} - V_{(2,G)}\right]$
8. DE/best/2/bin	$V_{(G+1)} = V_{(best,G)} + F\left[V_{(1,G)} + V_{(2,G)} - V_{(3,G)} - V_{(4,G)}\right]$
9. DE/rand/2/bin	$V_{(G+1)} = V_{(5,G)} + F\left[V_{(1,G)} + V_{(2,G)} - V_{(3,G)} - V_{(4,G)}\right]$

Crossover



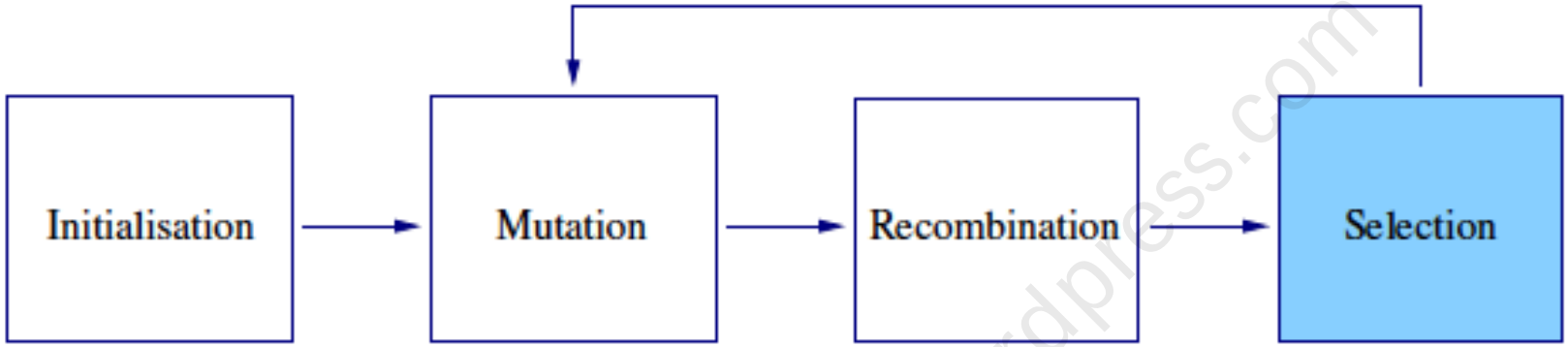
$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if } \text{rand}_{j,i} \leq CR \text{ or } j = \text{I}_{\text{rand}} \\ x_{j,i,G} & \text{if } \text{rand}_{j,i} > CR \text{ and } j \neq \text{I}_{\text{rand}} \end{cases}$$

$$i = 1, 2, \ldots, N; \; j = 1, 2, \ldots, D$$



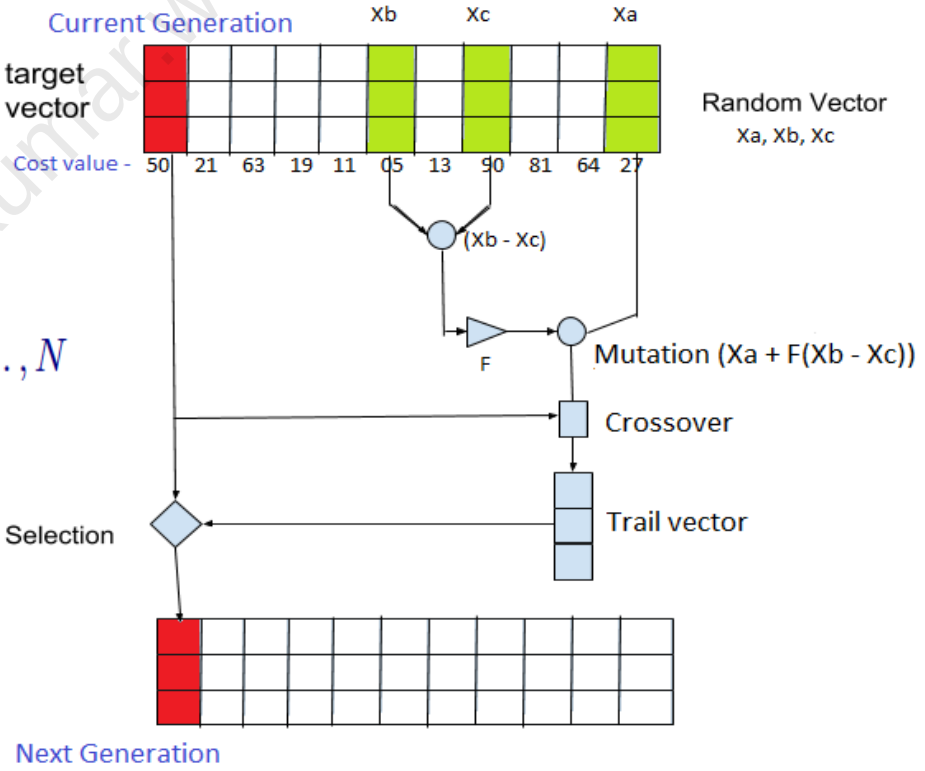
- The donor vector exchanges its components with the target vector X_i^G to form the trial vector.
- CR-Crossover Rate

Selection



- The fitness of target and trail vector is evaluated and the one with lowest function value is selected for next generation.

$$x_{i,G+1} = \begin{cases} u_{i,G+1} & \text{if } f(u_{i,G+1}) \le f(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases} \qquad i = 1, 2, \ldots, N$$



Pseudo Code

Algorithm 1. Pseudocode for classic Differential Evolution.

Input: Population size '*NP*', Problem Size 'D', Mutation Rate '*F*', Crossover Rate '*Cr*'; Stope_Criteria {Number of Generation, Target}, Upper Bound 'U', Lower Bound 'L'
Output: Best_Vector

```
Population = Initialize Population (NP, D, U, L);
While (Stope_Criteria ≠ True) do
    Best_Vector = EvaluatePopulation (Population);
    v_x = Select_Random_Vector (Population);
    Index = FindIndexOfVector (v_x); //specify row number of a vector
    Select_Random_Vector (Population,v_1, v_2, v_3) where v_1 ≠ v_2 ≠ v_3 ≠ v_x
  Vy = v_1, + F(v_2 − v_3)
        For (i = 0; i++; i < D − 1)//Loop for starting Crossover operation
            if (rand_j [0, 1) < CR)Then
            u[i] = v_x [i]
            Else u[i] = v_y [i]
        End For Loop//end crossover operation
    If (CostFunctionOfVector(u) ≤ CostFunctionOfVector (v_x)) Then
    UpdatePopulation (u, Index, Population);
  End; //While loop
Retune Best_Vector;
```

DE Working

Var	1	2	3	4	5	6
x1	0.68	0.92	0.22	0.12	0.40	0.94
x2	0.89	0.92	0.14	0.09	0.81	0.63
x3	0.04	0.33	0.40	0.05	0.83	0.13
f(x)	1.61	2.17	0.76	0.26	2.04	1.70

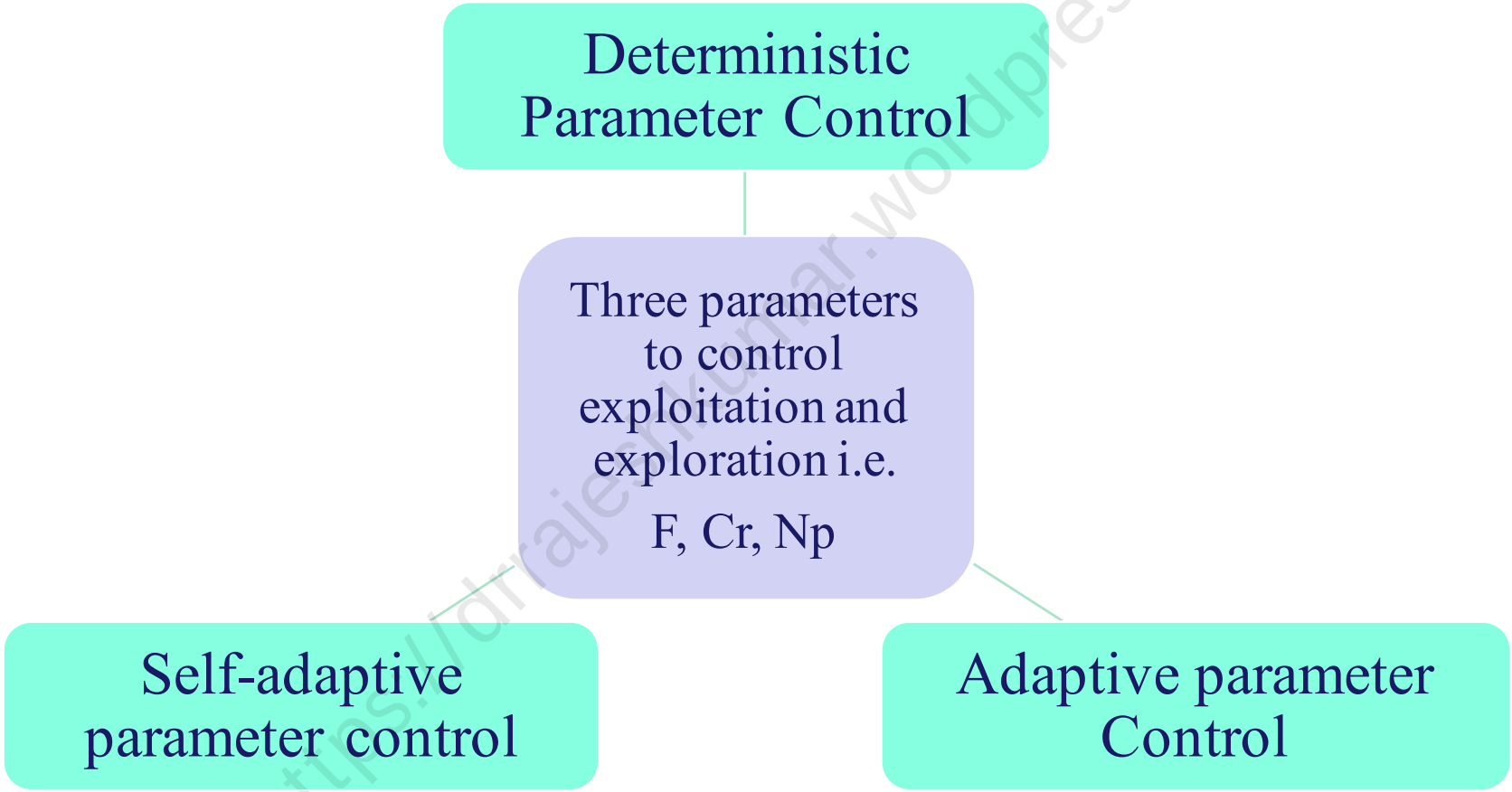
Var	2	4	Diff
x1	0.92	0.12	0.80
x2	0.92	0.09	0.83
x3	0.33	0.05	0.28

	Target Vector			Vector
x1	0.68		1.58	1.58
x2	0.89	CR = 0.5	1.29	0.89
x3	0.04		0.39	0.04

$$K = Diff * F(= 0.08)$$

K + noise(6)	
0.64 + 0.94	1.58
0.66 + 0.63	1.29
0.22 + 0.13	0.35

Scope to Create Variants of DE



Deterministic Parameter Control

Parameters are altered using a deterministic rule regardless of the feedback from the evolutionary search.

i. In first approach, F was created for each individual within the range (0.4, 1) and the interval (0.5, 0.7) was selected for Cr

ii. CoDE - Composite DE algorithm

CoDE

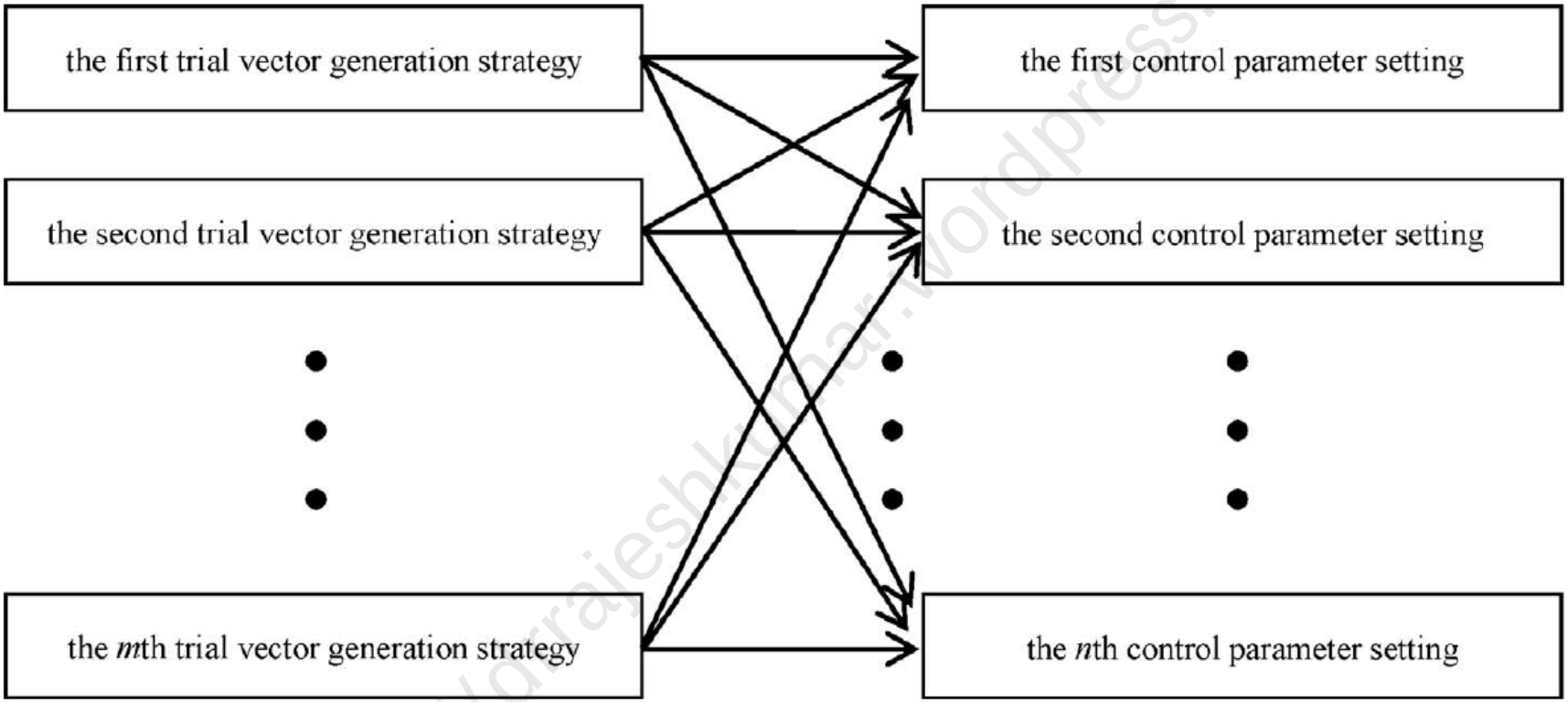
- A trial vector is selected from a set of groups.
- Main objective – To merge many trial vector strategies with different parameters at each iteration to construct new trial vectors.
- The selected strategies are:

1. DE/rand/1/bin
2. DE/rand/2/bin
3. DE/current-to- rand/1

- The three pair choices for the control parameter settings:

1. (F = 1.0, Cr = 0.1)
2. (F = 1.0; Cr = 0.9)
3. (F = 0.8; Cr = 0.2)

Illustration Of Combining Trial Vector Generation Strategies With Control Parameter Settings



Algo

Input: NP: the number of individuals at each generation, i.e., the population size.
Max_FES: maximum number of function evaluations.
the strategy candidate pool: "rand/1/bin", "rand/2/bin", and "current-to-rand/1".
the parameter candidate pool: [F=1.0, C_r=0.1], [F=1.0, C_r=0.9], and [F=0.8, C_r=0.2].

(1) G=0;

(2) Generate an initial population $P_0 = \{\vec{x}_{1,0},\ldots,\vec{x}_{NP,0}\}$ by uniformly and randomly sampling from the feasible solution space;

(3) Evaluate the objective function values $f(\vec{x}_{1,0}),\ldots,f(\vec{x}_{NP,0})$;

(4) FES=NP;

(5) **while** FES<Max_FES **do**

(6) $P_{G+1} = \emptyset$;

(7) **for** i=1:NP **do**

(8) Use the three trial vector generation strategies, each with a control parameter setting randomly selected from the parameter candidate pool, to generate three trial vectors $\vec{u}_{i\_1,G}$, $\vec{u}_{i\_2,G}$, and $\vec{u}_{i\_3,G}$ for the target vector $\vec{x}_{i,G}$;

(9) Evaluate the objective function values of the three trial vectors $\vec{u}_{i\_1,G}$, $\vec{u}_{i\_2,G}$, and $\vec{u}_{i\_3,G}$;

(10) Choose the best trial vector (denoted as $\vec{u}^*_{i,G}$) from the three trial vectors $\vec{u}_{i\_1,G}$, $\vec{u}_{i\_2,G}$, and $\vec{u}_{i\_3,G}$;

(11) $P_{G+1} = P_{G+1} \cup select(\vec{x}_{i,G}, \vec{u}^*_{i,G})$;

(12) FES=FES+3;

(13) **end for**

(14) G=G+1;

(15) **end while**

Output: the individual with the smallest objective function value in the population.

Adaptive Parameter Control

Applies an evaluation from feedback of the F relay on an additional parameter (γ) that it is necessary to be adjusted.

i. Differential Evolution with Self-Adapting Populations (DESAP)

ii. Fuzzy Adaptive Differential Evolution (FADE)

iii. Self-Adaptive Differential Evolution (SaDE)

iv. Self-Adaptive NSDE (SaNSDE)

v. Self-Adapting Parameter Setting in Differential Evolution (jDE)

vi. Adaptive DE Algorithm (ADE)

vii. Modified DE (MDE)

viii. Modified DE with P-Best Crossover (MDE_pBX)

ix. DE with Self-Adaptive Mutation and Crossover (DESAMC)

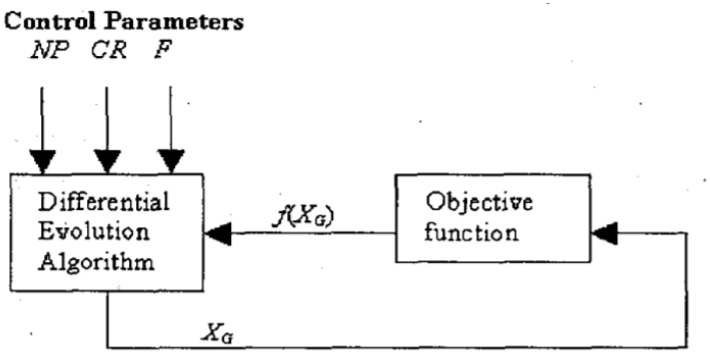
x. Adaptive Differential Evolution with Optional External Archive (JADE)

xi. Differential Covariance Matrix Adaptation Evolutionary Algorithm (CMA-ES)

Differential Evolution With Self-adapting Populations (DESAP)

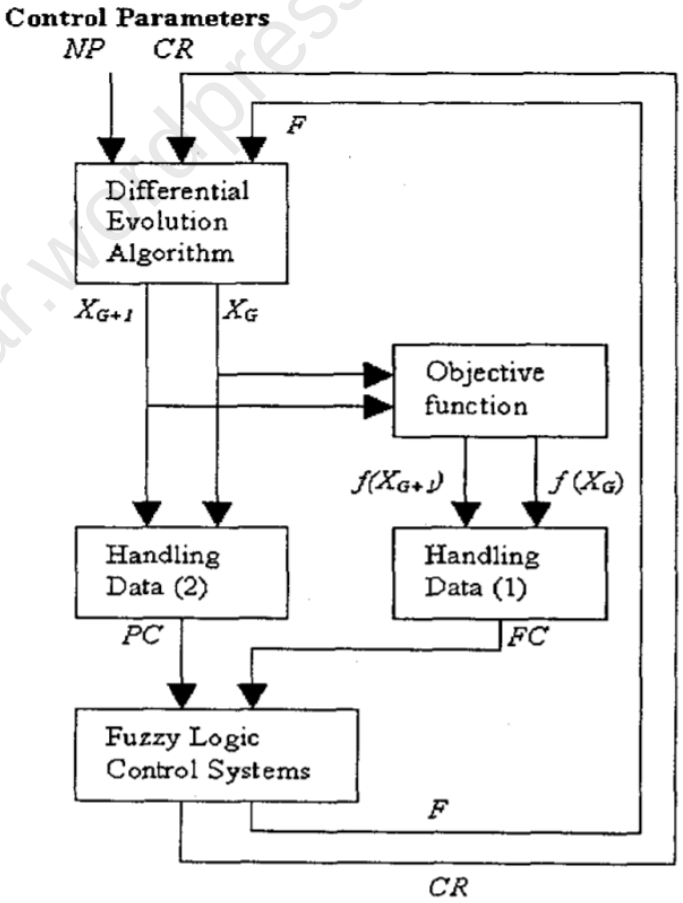
- DESAP dynamically adjusts the crossover and mutation parameters δ, η and the population size π.
- DESAP performed better than DE in one of De Jong's five exam problems, whereas the other solutions are almost identical.
- The mutation factor F is retained as static, and η denotes the probability of implementing an extra mutation operation by using normally distributed.

Fuzzy Adaptive Differential Evolution (FADE)



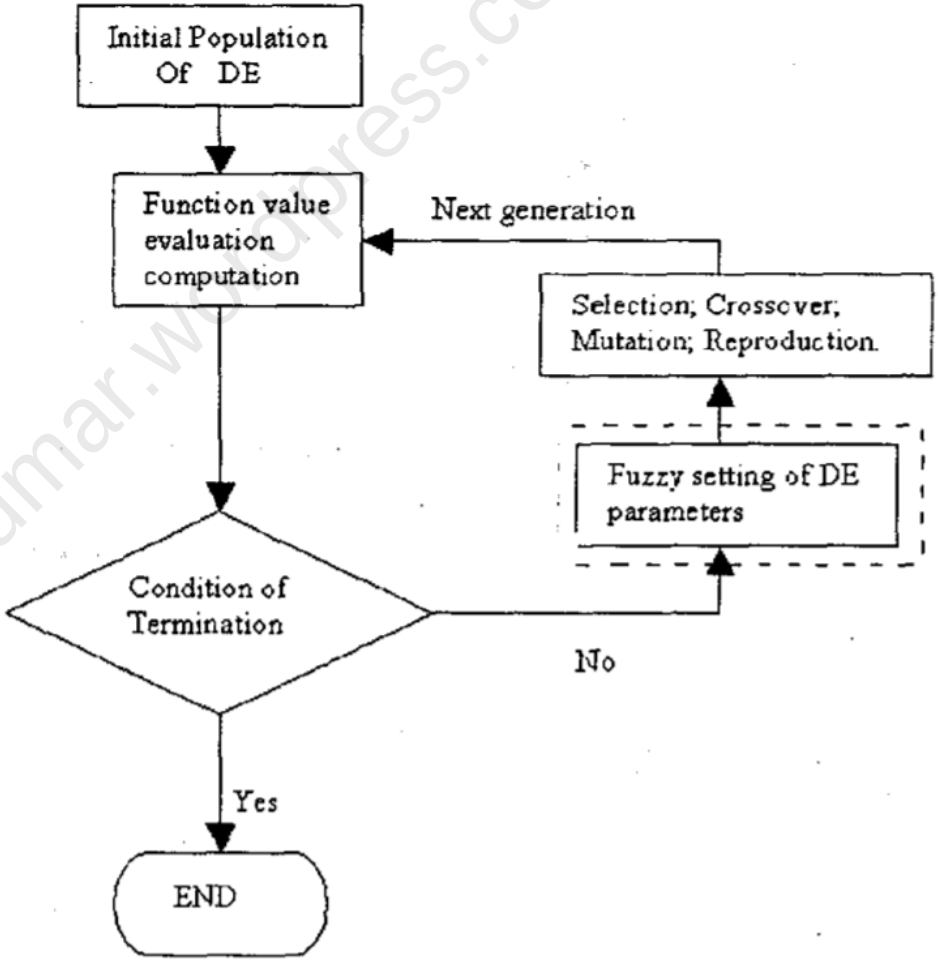
Basic DE

FADE



FADE

- The purpose of using fuzzy logic is to make the DE'S control parameters adaptive in the minimization process.
- To minimize the error(e) between the global optimum and the actual function value in case of knowing about global optimum value.
- Minimize the change in e of function values between successive generation in case of knowing nothing about global optimum function value.



Self-adaptive differential evolution (SaDE)

- Self-adaptive differential evolution (SaDE) is simultaneously applied to a pair of mutation techniques "DE/rand/1" and "DE/current-to-best/2"
- In SADE, a mutation strategy is selected probabilistically.

- The 2 mutation startegies used are:

  - "DE/rand/1" : $v_i = x_{R1} + F(x_{R2} - x_{R3})$
  - "DE/current-to-best/1" : $v_i = x_{R1} + F(x_{best} - x_i) + F(x_{R1} - x_{R2})$

- Probability of one is p_1 and another is $p_2=(1-p_1)$.

SaDE

For population size = NP, randomly generate a vector of size NP with uniform distribution in the range [0, 1] for each element.

If j^{th} element of vector <= p1 then *DE/rand/1* will be applied to the j^{th} individual of population else *DE/current-to-best/1* is applied.

All trial vectors are evaluated. ns1, ns2 are successful trail vectors for both startegies respectively. ns1, ns2 are successful trail vectors for both startegies respectively.

$$p_1 = \frac{ns1.(ns2 + nf2)}{ns2.(ns1 + nf1) + ns1(ns2 + nf2)} \quad \& \quad p_2 = 1 - p_1$$

$F_i = NR(0.5, 0.3)$ and $Cr_i = NR(Cr_m, 0.1)$

Cr and F are created based on normal distributions for each generation.

Self-adaptive NSDE (SaNSDE)

- Same as basic DE except mutation strategy i.e.

$$Vi = xr3 + \begin{cases} (xr1 - xr2) * N(0.5,0.5) & if\ u(0.1)0.5 \\ (xr1 - xr2) * \delta & otherwise \end{cases}$$

- Different than SaDE in terms that F and Cr are created on the basis of Cauchy distribution rather than normal distribution.
- In SaDE, two DE learning strategies are chosen according to their performance.
- The most appropriate learning technique and parameter values are increasingly self-adapted according to the learning experience gained during evolution

Self-Adapting Parameter Setting in Differential Evolution (jDE)

- Improves the population size throughout the optimization process based on the improved parameters.
- Generates vectors that are more likely to survive.
- F_i and Cr_i are assigned and adapted for each individual.
- Initially F_i and Cr_i are 0.5 and 0.9 respectively.

$$F_{i,g} = \begin{cases} 0.1 + 0.9 * random1, if\ random2 < \tau_1 \\ F_{i,g}\ otherwise \end{cases}$$

$$Cr_{i,g} = \begin{cases} random3, if\ random4 < \tau_2 \\ Cr_{i,g}\ otherwise \end{cases}$$

- where, *random j* = 1, 2, 3, 4 is the uniform random function ϵ[0, 1]

Adaptive DE Algorithm (ADE)

- In this, F and Cr are modified to each iteration using the current generation and the fitness.
- The mutation and crossover operations are calculated for each generation.
- For each parent of generation $g = x_i^g$

$$Cr(G) = Cr_0 * \left(\frac{\exp(I_G) - \exp(-I_G)}{40} + 1\right)$$

$$Cr(G) = Cr_0 * \left(\frac{\exp(I_G) - \exp(-I_G)}{40} + 1\right)$$

$$I_G = 3 - 6 * (\frac{G}{Gm})$$

Modified DE (MDE)

- MDE uses only one array.
- Array is updated when a better solution is found.
- F – adjusted by Laplace distribution.
- Improved the convergence speed and fewer evaluation procedures than basic DE.

Modified DE with p-best crossover (MDE_pBX)

- F and Cr produced using Cauchy distribution.
- Mutation strategy used:

$$V_{i,g} = X_{i,g} + F_i(X_{grbest} - X_{i,g} + X_{r1,g} - X_{r2,g})$$

- p-best crossover : Random vector is chosen from best p vectors and crossover is done.

$$p = ceil\left[\frac{Np}{2}.(1 - \frac{G-1}{Gmax})\right]$$

G = present generation
Gmax= max no. of generations

DE with self-adaptive mutation and crossover (DESAMC)

- F is adapted using an affection index (A*fi*).

$$F_i(g) = \frac{1}{1 + \tanh(2Af_i(g))}$$

$$Cr_i(g) = Cr^- + (1 - \frac{g}{gmax})^{\frac{g}{gmax}}(Cr^+ - Cr^-)$$

- Cr+ and Cr- are the maximum and minimum values of Cr.

Adaptive differential evolution with optional external archive (JADE)

The crossover and selection operations are implemented as in the classic DE algorithm.

The greedy strategy involves a new mutation strategy called DE/current-to-pbest/1 (without archives) and assists the baseline JADE:

$$V_{i,g} = X_{i,g} + F_i\,(V_{best,\,g} - V_{1,g}) + F_i\,(V_{2,g} - V_{3,g})$$

$$V_{i,g} = X_{i,g} + F_i\,(V_{best,\,g} - V_{1,g}) + F_i\,(V_{2,g} - V'_{3,G})$$

where $V_{best,g}$ is the best solution that is randomly chosen as one of the best individuals from the current population [56]. Similarly, $V_{1,g}$, $V_{2,g}$ and $V_{3,g}$ are randomly selected from the current population. However, $V'_{3,G}$ is also randomly chosen from the union between $X_{i,g}$ and $V_{1,g}$.

$$V'_{3,G} = \text{randomly}\ (V_{1,g} \cup X_{i,g})$$

Differential Evolution with Multiple Strategies

- Four mutation strategies are selected to form a pool.
- One crossover operator is used.
- For each individual mutation strategy is selected randomly from the pool.

DE-PSO

- Process of DE-PSO is as follows:

```
Initialize population(size=N)
For i = 1 to N do
Select r1,r2,r3 ∈ N
        For j = 1 to D do
        Select j_rand ∈ D
        Mutation and Crossover (basic DE)
        If (fitness(trail vector)>fitness(target vector))
                Selection Process(Basic DE)
        Else
                PSO Activated (New particle is found using PSO)
Again Selection is done as Basic DE
```

Other Hybrid Approaches

Methods hybridized	Authors	Optimization problems applied to
DE + Artificial Bee Colony (ABC) Algorithm	Tran et al.	Multi-objective optimization for optimal time-cost-quality trade-off for planning of construction projects
DE + Ant Colony Optimization (ACO)	Chang et al.	Single-objective real parameter engineering optimization
DE + Bacterial Foraging based Optimization (BFO)	Biswal et al.	Single-objective continuous parameter clustering problem
DE + Gravitational Search Algorithm (GSA)	Chakraborti et al.	Binary optimization problem involving highly discriminative feature subset selection
DE + Invasive Weed Optimization (IWO)	Basak et al.	Single-objective, bound constrained function optimization problems
DE + Firefly Algorithm(FFA)	Abdullah et al.	Single-objective nonlinear optimization problem involving estimation of biological model parameters
DE + Fireworks Algorithm (FWA)	Zheng et al.	Single-objective, bound constrained function optimization problems

Thank You