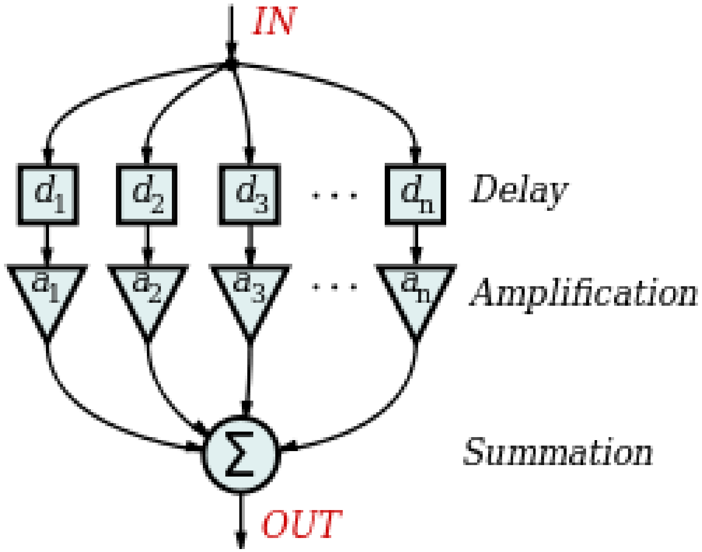
Differential evolution

Differential evolution

- Differential evolution (DE) was developed by Rainer Storn and Kenneth V. Price around 1995.
- DE is a unique evolutionary algorithm because it is **not biologically motivate**
- DE is used for multidimensional real-valued functions but **does not use the gradient** of the problem being optimized, which means DE **does not require** the optimization problem to be **differentiable**.
- DE can therefore also be used on optimization problems that are **not even continuous**, are noisy, change over time, etc.

Differential evolution

- Inspired from the real worlds problem of **Digital filter coefficients**
- In signal processing, a **digital filter** is a system that performs mathematical operations on a sampled, discrete-time signal to **reduce or enhance certain aspects of that signal**.



E=w(Exp-Actual)

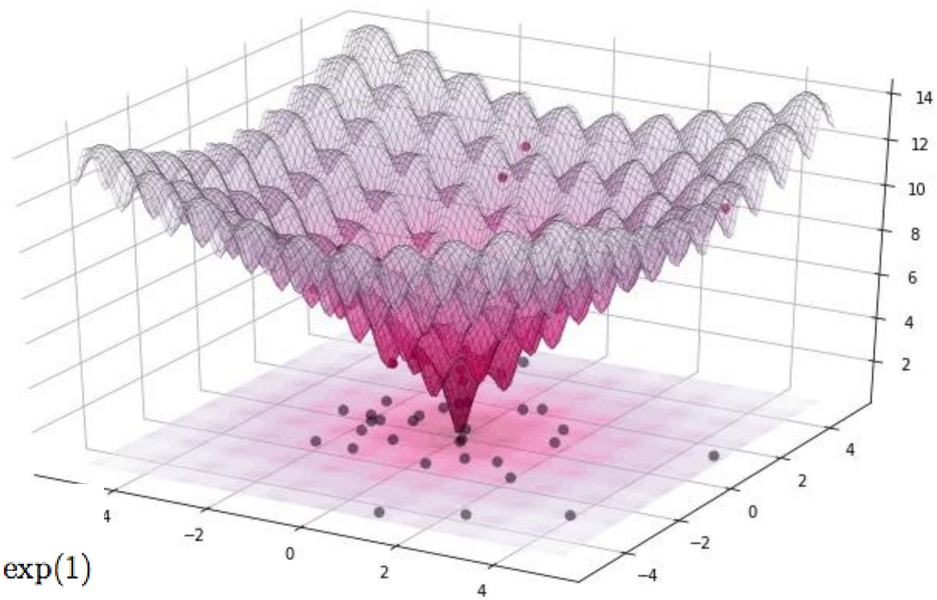
Differential evolution

- DE optimizes a problem by maintaining a **population of candidate solutions** and **creating new candidate solutions** by **combining existing ones** according to its **simple formulae**, and then **keeping** whichever candidate solution has the **best score or fitness** on the optimization problem at hand.

Global Minimum:

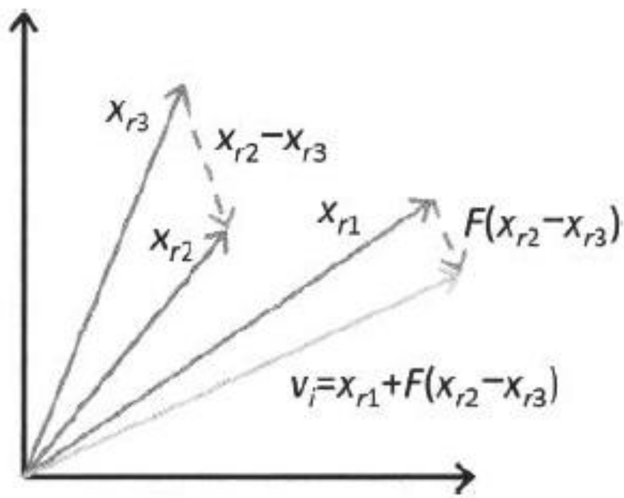
$$f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (0, \dots, 0)$$

$x_i \in [-32.768, 32.768]$



$$f(\mathbf{x}) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

A BASIC DIFFERENTIAL EVOLUTION ALGORITHM



$$u_{ij} = \begin{cases} v_{ij} & \text{if } (r_{cj} < c) \text{ or } (j = \mathcal{J}_r) \\ x_{ij} & \text{otherwise} \end{cases}$$

- DE is based on the idea of **taking the difference vector** between two individuals, and **adding a scaled version** of the difference vector to a **third individual** to create a new candidate solution.

A BASIC DIFFERENTIAL EVOLUTION ALGORITHM

F = stepsize parameter $\in [0.4, 0.9]$
c = crossover rate $\in [0.1, 1]$
Initialize a population of candidate solutions $\{x_i\}$ for $i \in [1, N]$
While not(termination criterion)
    For each individual x_i, $i \in [1, N]$
        $r_1 \leftarrow$ random integer $\in [1, N] : r_1 \neq i$
        $r_2 \leftarrow$ random integer $\in [1, N] : r_2 \notin \{i, r_1\}$
        $r_3 \leftarrow$ random integer $\in [1, N] : r_3 \notin \{i, r_1, r_2\}$
        $v_i \leftarrow x_{r1} + F(x_{r2} - x_{r3})$ (mutant vector)
        $\mathcal{J}_r \leftarrow$ random integer $\in [1, n]$
        For each dimension $j \in [1, n]$
            $r_{cj} \leftarrow$ random number $\in [0, 1]$
            If $(r_{cj} < c)$ or $(j = \mathcal{J}_r)$ then
                $u_{ij} \leftarrow v_{ij}$
            else
                $u_{ij} \leftarrow x_{ij}$
            End if
        Next dimension
    Next individual
    For each population index $i \in [1, N]$
        If $f(u_i) < f(x_i)$ then $x_i \leftarrow u_i$
    Next population index
Next generation

A BASIC DIFFERENTIAL EVOLUTION ALGORITHM

- This algorithm is often referred to as **classic DE**.
- It is also called **DE/rand/1/bin** because the base vector, x_{ri}, is **randomly chosen**; **one vector difference** (that is, $F(x_{r2} - x_{r3})$) is added to x_{r1} and the number of mutant vector elements that are contributed to the trial vector closely follows a **binomial distribution**.
- In probability theory and statistics, the **binomial distribution** with parameters *n* and *p* is the discrete probability distribution of the number of successes in a sequence of *n* independent experiments, each asking a yes–no question, and each with its own Boolean-valued outcome: yes or no (with probability $q = 1 - p$).
- It would exactly follow a binomial distribution if not for the "$j = J_r$" test

DIFFERENTIAL EVOLUTION VARIATIONS

- **Trial Vectors**
  - **DE/rand/1/L** works by generating a random integer L ϵ [l,n], copying L consecutive features from vi to ui, and then copying the remaining features from Xi to Ui

$L \leftarrow$ random integer $\in [1, n]$
$s \leftarrow$ random integer $\in [1, n]$
$J \leftarrow \{s, \min(n, s + L - 1)\} \cup \{1, s + L - n - 1\}$
For each dimension $j \in [1, n]$
    If $j \in J$
        $u_{ij} \leftarrow v_{ij}$
    else
        $u_{ij} \leftarrow x_{ij}$
    End if
Next dimension

DIFFERENTIAL EVOLUTION VARIATIONS

- For example, suppose that we have a seven-dimensional problem ($n = 7$). The DE/rand/1/L algorithm works by first generating a random integer $L \in [l,n]$; suppose that **L = 3**. We then generate a random starting point $s \in [l,n]$; suppose that **s = 6**.

$$
\begin{aligned}
u_{i1} &\leftarrow v_{i1} \\
u_{i2} &\leftarrow x_{i2} \\
u_{i3} &\leftarrow x_{i3} \\
u_{i4} &\leftarrow x_{i4} \\
u_{i5} &\leftarrow x_{i5} \text{ (ending point)} \\
u_{i6} &\leftarrow v_{i6} \text{ (starting point } s\text{)} \\
u_{i7} &\leftarrow v_{i7}.
\end{aligned}
$$

DIFFERENTIAL EVOLUTION VARIATIONS

$$E(\text{number of } v_i \text{ elements copied}) = 1 + c(n-1) \qquad \text{for DE/rand/1/bin.}$$

$$E(\text{number of } v_i \text{ elements copied}) = n/2 \qquad \text{for DE/rand/1/L.}$$

Under what conditions is the expected number of mutant vector elements copied to the trial vector equal for the bin and L options?

DIFFERENTIAL EVOLUTION VARIATIONS

$$E(\text{number of } v_i \text{ elements copied}) = 1 + c(n-1) \qquad \text{for DE/rand/1/bin.}$$

$$E(\text{number of } v_i \text{ elements copied}) = n/2 \qquad \text{for DE/rand/1/L.}$$

Under what conditions is the expected number of mutant vector elements copied to the trial vector equal for the bin and L options?

$$c = \frac{n-2}{2(n-1)}.$$

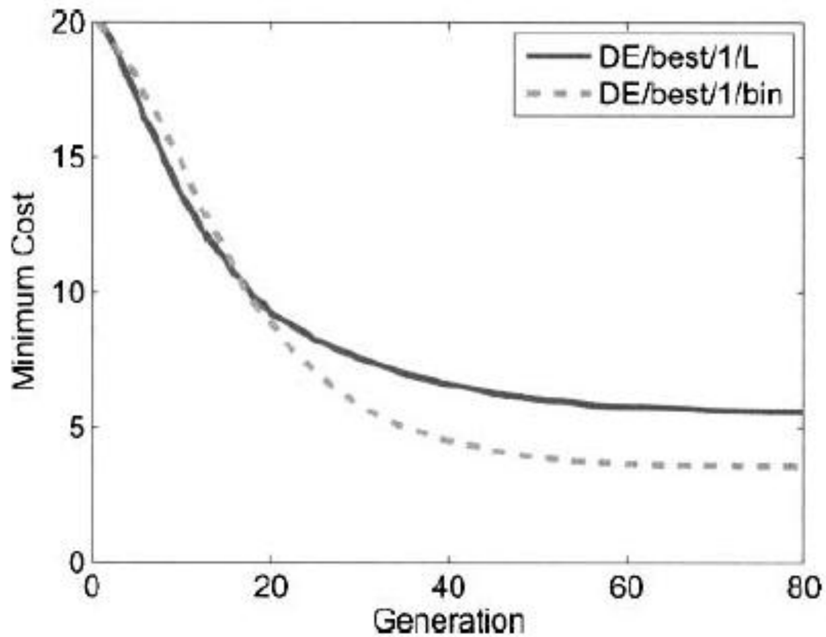
Mutant Vectors

- Instead of randomly choosing the base vector x_{r1} , it may be beneficial to always **use the best individual** in the population **as the base vector**.
- That way the entire set of trial vectors U_i for $i \in [1, n]$ is comprised of mutations of the **best individual**.
- This approach is called **DE/best/1/bin.**

$$v_i \leftarrow x_b + F(x_{r2} - x_{r3})$$

- where x_b is the best individual in the population.

Mutant Vectors



DE performance on the 20-dimensional Ackley function

Mutant Vectors

- Another option is to **use two difference vectors** to create the mutant vector [Storn and sPrice, 1996]

$$
\begin{aligned}
r_4 &\leftarrow \text{random integer} \in [1, N] : r_4 \notin \{i, r_1, r_2, r_3\} \\
r_5 &\leftarrow \text{random integer} \in [1, N] : r_5 \notin \{i, r_1, r_2, r_3, r_4\} \\
v_i &\leftarrow \begin{cases} x_{r1} + F(x_{r2} - x_{r3} + x_{r4} - x_{r5}) & \text{DE/rand/2/?} \\ x_b + F(x_{r2} - x_{r3} + x_{r4} - x_{r5}) & \text{DE/best/2/?} \end{cases}
\end{aligned}
$$

- DE/rand/2/bin or DE/best/2/bin
- DE/rand/2/L or DE/best/2/L

Mutant Vectors

- DE can also be implemented by using the current X_i as the base vector

$$v_i \leftarrow x_i + F\Delta x$$

where Δx is a difference vector.

  - DE/target/1/bin, DE/target/2/bin,
  - DE/target/1/L, or DE/target/2/L

Mutant Vectors

- Yet another option is to create the difference vector by **using the best individual** in the population, x_b.
- This tends to create mutant vectors that all **move toward x_b.** The vector that is subtracted from x_b could be a random individual or the base individual.

$$
\begin{aligned}
v_i &\leftarrow x_i + F(x_b - x_i) \\
v_i &\leftarrow x_{r1} + F(x_b - x_{r3}) \\
v_i &\leftarrow x_b + F(x_{r2} - x_{r3} + x_b - x_{r5}) \\
v_i &\leftarrow x_i + F(x_b - x_i + x_{r2} - x_{r3})
\end{aligned}
$$

- If the last equation above is used to generate Vi,

the algorithm is called **DE/target-to-best/1/bin**

Mutant Vectors

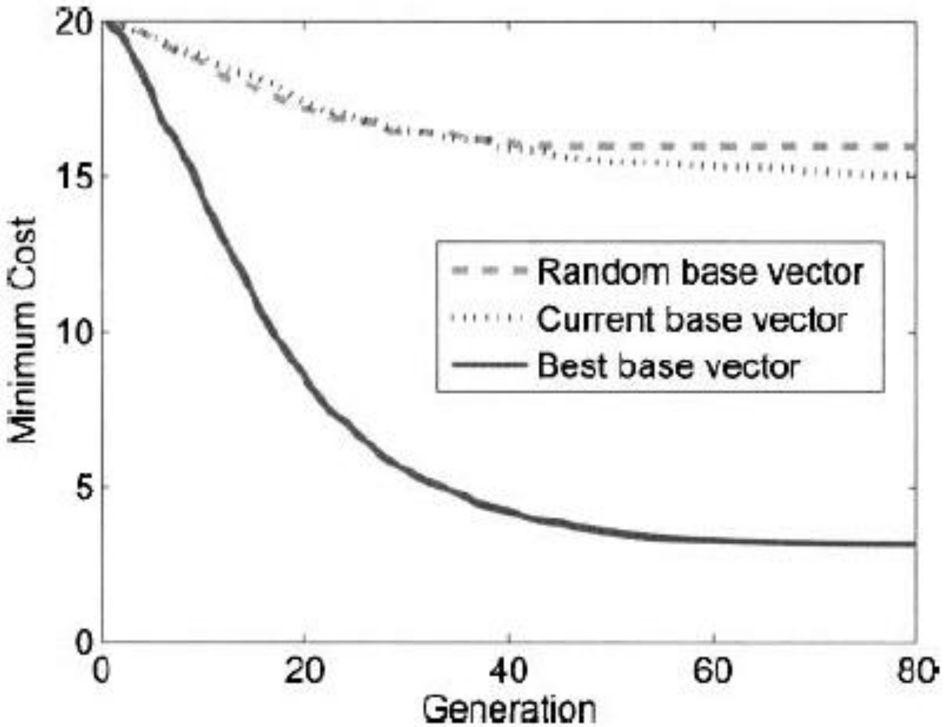
- **either-or algorithm**
- We could **combine various methods** by randomly deciding how to generate the mutant vector.

p_f = mutation probability $\in [0, 1]$
$a \leftarrow$ random number $\in [0, 1]$
If $a < p_f$ then
$\quad v_i \leftarrow x_{r1} + F(x_{r2} - x_{r3})$
else
$\quad v_i \leftarrow x_{r1} + K(x_{r2} - x_{r1} + x_{r3} - x_{r1})$
End if

Mutant vector generation that results in the DE/rand/1/either-or algorithm.

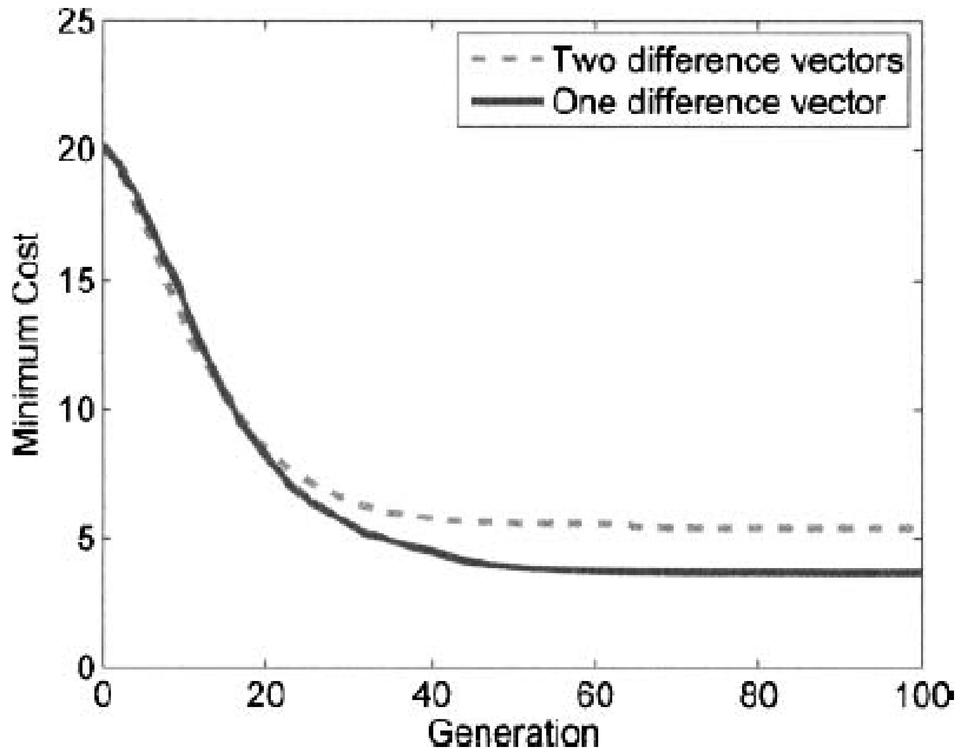
- **K = (F + l)/2** gives good results in benchmark problems.

Mutant Vectors



DE performance on the 20-dimensional Ackley function.

Mutant Vectors



DE performance on the 20-dimensional Ackley function.

Scale Factor Adjustment

- DE's **scale factor F** determines the effect that difference vectors have on the mutant vector.
- So far we have assumed that F is a constant.
- We can vary the DE scale factor two different ways.
  - Dither
  - Jitter
- **Dither:-** we can allow F to **remain a scalar** and randomly change it each time through the "for each individual“
- **Jitter**:-we can change **F to an n-element vector** and randomly change each element of F in the "for each individual" loop, so that each element of the mutant vector v is modified by a uniquely-scaled component of the difference vector.

Scale Factor Adjustment

Dither

$$
\begin{aligned}
F &\leftarrow U[F_{\min}, F_{\max}] \\
v_i &\leftarrow x_{r1} + F(x_{r2} - x_{r3}).
\end{aligned}
$$

Jitter

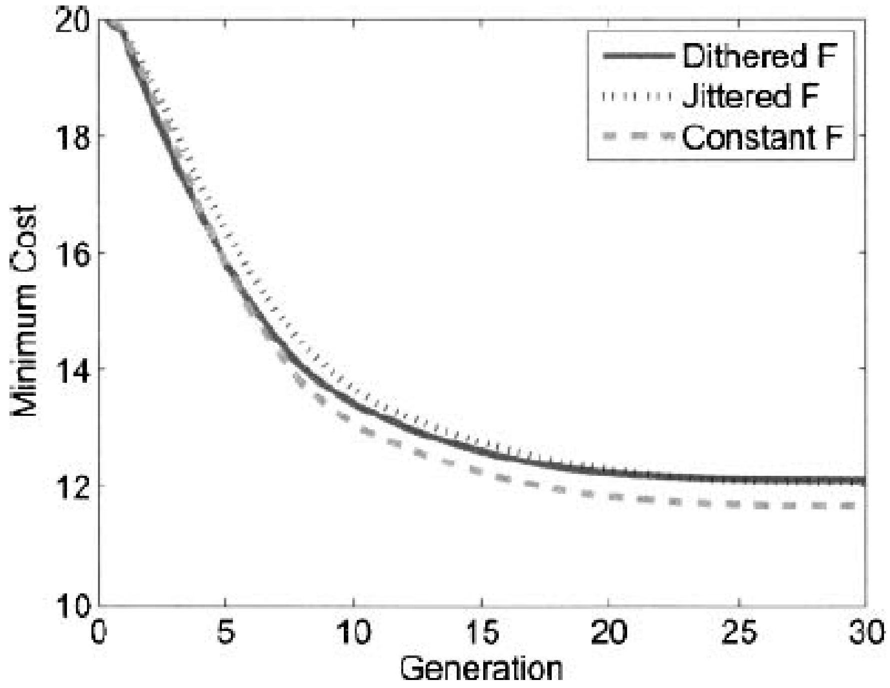
For each dimension $j \in [1, n]$

$$F_j \leftarrow U[F_{\min}, F_{\max}]$$

$$v_{ij} \leftarrow x_{r1,j} + F_j(x_{r2,j} - x_{r3,j})$$

Next dimension

Scale Factor Adjustment



- DE performance on the 20-dimensional Ackley function with crossover rate c = 0.9. The traces show the cost of the best individual at each generation, averaged over 100 Monte Carlo simulations. The use of a constant scale factor F performs slightly better than dithering or jittering.

DISCRETE OPTIMIZATION

- The only place that discrete domains cause a problem in DE is in the generation of the **mutant vector**.

$$v_i \leftarrow x_{r1} + F(x_{r2} - x_{r3}).$$

- Since F ∊ [0,1], **V**$_i$ **might not** belong the problem **domain D**.

Mixed-Integer Differential Evolution

- One obvious approach to ensure that $V_i \in D$ is to simply project it onto D.
- For example, if D is the set of n-dimensional integer vectors, then

$$v_i \leftarrow \text{round}[x_{r1} + F(x_{r2} - x_{r3})]$$

round function operates element-by-element on a vector

- A more general way to do this is

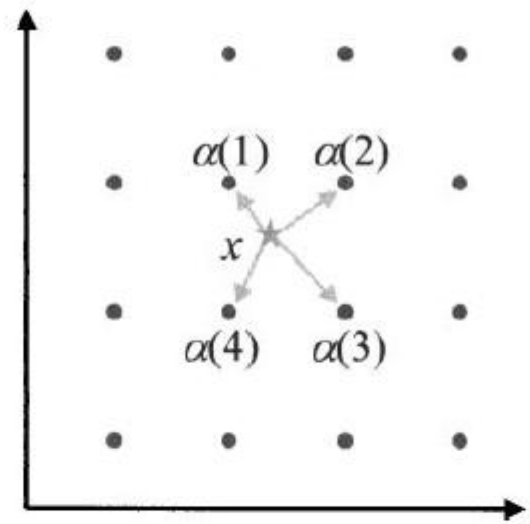
$$v_i \leftarrow P[x_{r1} + F(x_{r2} - x_{r3})]$$

where P is a projection operator such that **P(x) ϵ D** for all x.

Mixed-Integer Differential Evolution

- P could be more complicated

$$P(x) = \arg\min_{\alpha} f(\alpha) : \alpha \in D, |x_j - \alpha_j| < 1 \text{ for all } j \in [1, n].$$



Projection of the continuous-valued vector *x onto a discrete-valued vector a.*

Discrete Differential Evolution

- Another way to modify DE for discrete problems is to **change the mutant vector generation method** so that it directly creates mutant vectors that lie in the discrete domain D.

$$v_i \leftarrow G(x_{r1}, x_{r2}, x_{r3})$$

$$v_i \quad \leftarrow \quad x_{r1} + \mathrm{round}[F(x_{r2} - x_{r3})]$$

$$v_i \quad \leftarrow \quad x_{r1} + \mathrm{sign}(x_{r2} - x_{r3})$$

DIFFERENTIAL EVOLUTION AND GENETIC ALGORITHMS

Initialize a population of candidate solutions $\{x_i\}$, $i \in [1, N]$
While not(termination criterion)
    For each individual x_i, $i \in [1, N]$
        $r_1 \leftarrow$ random integer $\in [1, N] : r_1 \neq i$
        $v_i \leftarrow x_{r1}$
        For each dimension $j \in [1, n]$
            If rand(0,1) $< c$ then
                $u_{ij} \leftarrow v_{ij}$
            else
                $u_{ij} \leftarrow x_{ij}$
            End if
        Next dimension
    Next individual
    For each $i \in [1, N]$, If $f(u_i) < f(x_i)$ then $x_i \leftarrow u_i$
Next generation

DIFFERENTIAL EVOLUTION AND GENETIC ALGORITHMS

Initialize a population of candidate solutions $\{x_i\}$, $i \in [1, N]$
While not(termination criterion)
    For each individual x_i, $i \in [1, N]$
        $r_1 \leftarrow$ random integer $\in [1, N] : r_1 \neq i$
        $r_2 \leftarrow$ random integer $\in [1, N] : r_2 \notin \{i, r_1\}$
        $r_3 \leftarrow$ random integer $\in [1, N] : r_3 \notin \{i, r_1, r_2\}$
        $v_i \leftarrow x_{r1} + F(x_{r2} - x_{r3})$
        $\mathcal{J}_r \leftarrow$ random integer $\in [1, n]$
        For each dimension $j \in [1, n]$
            If ($\texttt{rand}(0,1) < c$) or ($j = \mathcal{J}_r$) then
                $u_{ij} \leftarrow v_{ij}$
            else
                $u_{ij} \leftarrow x_{ij}$
            End if
        Next dimension
    Next individual
    For each $i \in [1, N]$, If $f(u_i) < f(x_i)$ then $x_i \leftarrow u_i$
Next generation

Thank you