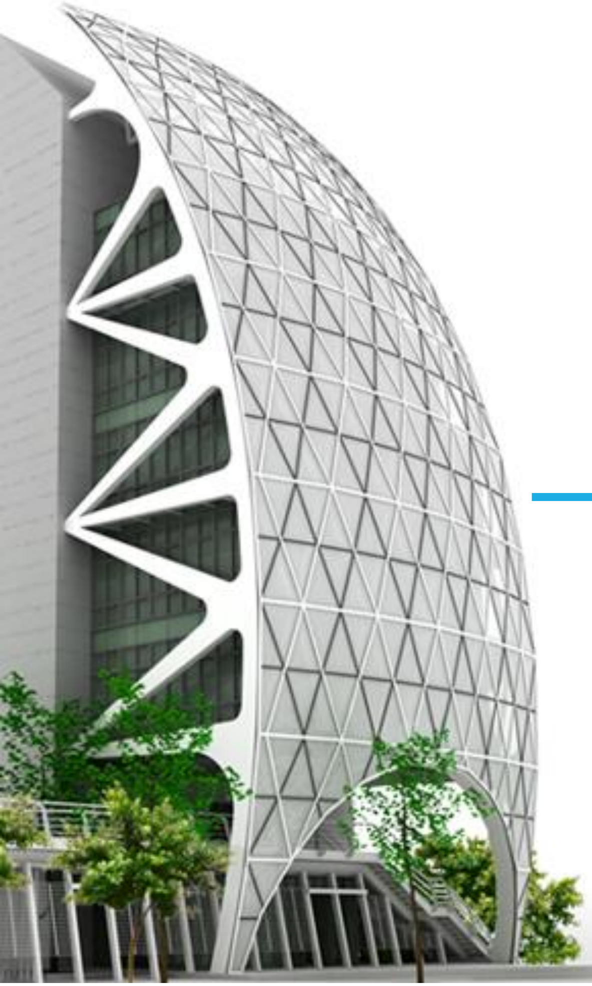

# IM5110701 Metaheuristics

Dr. Hendri Sutrisno

Institute of Statistical Science

Academia Sinica

Dr. Chao-Lung Yang

Department of Industrial Management

National Taiwan University of Science and Technology

# Chapter 4: Differential Evolution

Sean Luke, Essentials of Metaheuristics, Second Edition

# Initial work

- Differential Evolution (DE) was proposed by Kenneth Price and Rainer Storn [1] in 1997.
- Designed to solve problem with real-valued search space.
  - DE might not be suitable for Discrete Optimization
- Compared to other Evolutionary-based metaheuristics, DE has two distinct “twist” strategies.

[1] Storn, R., & Price, K.V. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization, 11*, 341-359.

# The tweaks

1. “Children” must compete directly against their immediate “parents” for inclusion in the population.
2. DE determines the size of Mutates largely based on the current variance in the population.
   - If the population is spread out, Mutate will make major changes.
   - If the population is condensed in a certain region, Mutates will be small.

# Pseudocode

**Algorithm 38** *Differential Evolution (DE)*

```
α ← mutation rate                         ▷ Commonly between 0.5 and 1.0, higher is more explorative
popsize ← desired population size

P ← ⟨ ⟩          ▷ Empty population (it's convenient here to treat it as a vector), of length popsize
Q ← □                     ▷ The parents. Each parent Q_i was responsible for creating the child P_i
for i from 1 to popsize do
    P_i ← new random individual
Best ← □
repeat
    for each individual P_i ∈ P do
        AssessFitness(P_i)
        if Q ≠ □ and Fitness(Q_i) > Fitness(P_i) then
            P_i ← Q_i                                  ▷ Retain the parent, throw away the kid
        if Best = □ or Fitness(P_i) > Fitness(Best) then
            Best ← P_i
    Q ← P
    for each individual Q_i ∈ Q do                     ▷ We treat individuals as vectors below
        a⃗ ← a copy of an individual other than Q_i, chosen at random with replacement from Q
        b⃗ ← a copy of an individual other than Q_i or a⃗, chosen at random with replacement from Q
        c⃗ ← a copy of an individual other than Q_i, a⃗, or b⃗, chosen at random with replacement from Q
        d⃗ ← a⃗ + α(b⃗ − c⃗)                                ▷ Mutation is just vector arithmetic
        P_i ← one child from Crossover(d⃗, Copy(Q_i))
until Best is the ideal solution or we ran out of time
return Best
```

Tweak #1

Tweak #2

# Mutation strategy

- Unlike the mutation in other Evolutionary-based metaheuristics, DE's mutation operators employ vector addition and subtraction
  - Designed for metric vector spaces (booleans, metric integer spaces, reals)
- To mutate a solution
  - DE generates a new child by picking three individuals from the population and doing the vector processes (Line #20 in the Pseudocode)

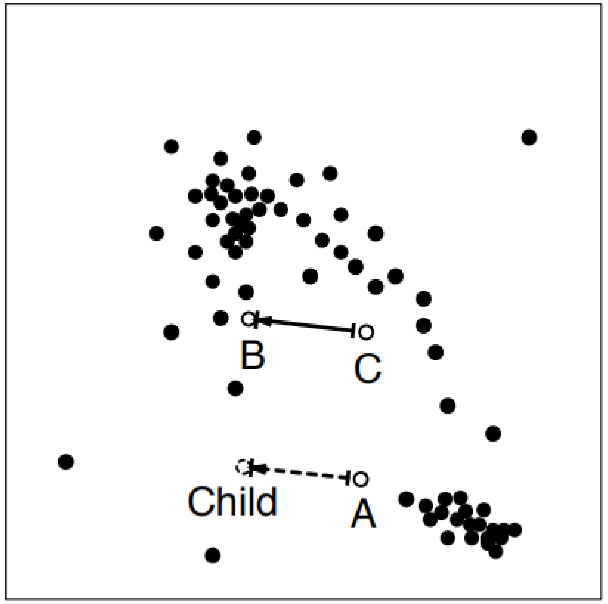


*Figure 15* Differential Evolution's primary mutation operator. A copy of individual $A$ is mutated by adding to it the vector between two other individuals $B$ and $C$, producing a child.

# Common terms in DE

- Target vector
  - The old solution to be replaced by the newly generated solution
  - In other evolutionary-based metaheuristics, target vector is also known as "Parent"
- Mutant vector or donor vector
  - The generated solution based on the some vector calculations in DE
  - Provide some information that can be used to create the trial vector
- Trial vector
  - The new candidate solution, created to replace the target vector
  - Obtained through crossover operation between target and mutant vectors

# Vector additions and subtractions to generate the mutant vector

- For each member of the population
  - Select three other solutions ($\vec{a}, \vec{b}, \vec{c}$)
  - Generate the mutant vector ($\vec{d}$)

$$\vec{d} \leftarrow \vec{a} + \alpha(\vec{b} - \vec{c})$$

  - $\alpha$ is the mutation rate, usually a real number between [0,2]. Higher $\alpha$ means higher explorative ability. $\alpha$ is determined by the user

# Crossover in DE

- The crossover constant ($CR$) control the probability of crossover
- $CR$ is determined by the user
- For each vector unit, we generate a random number.
  - **IF** the random number is lower than $CR$; pass the vector unit from mutant vector to trial vector
  - **ELSE;** pass the vector unit from target vector to trial vector

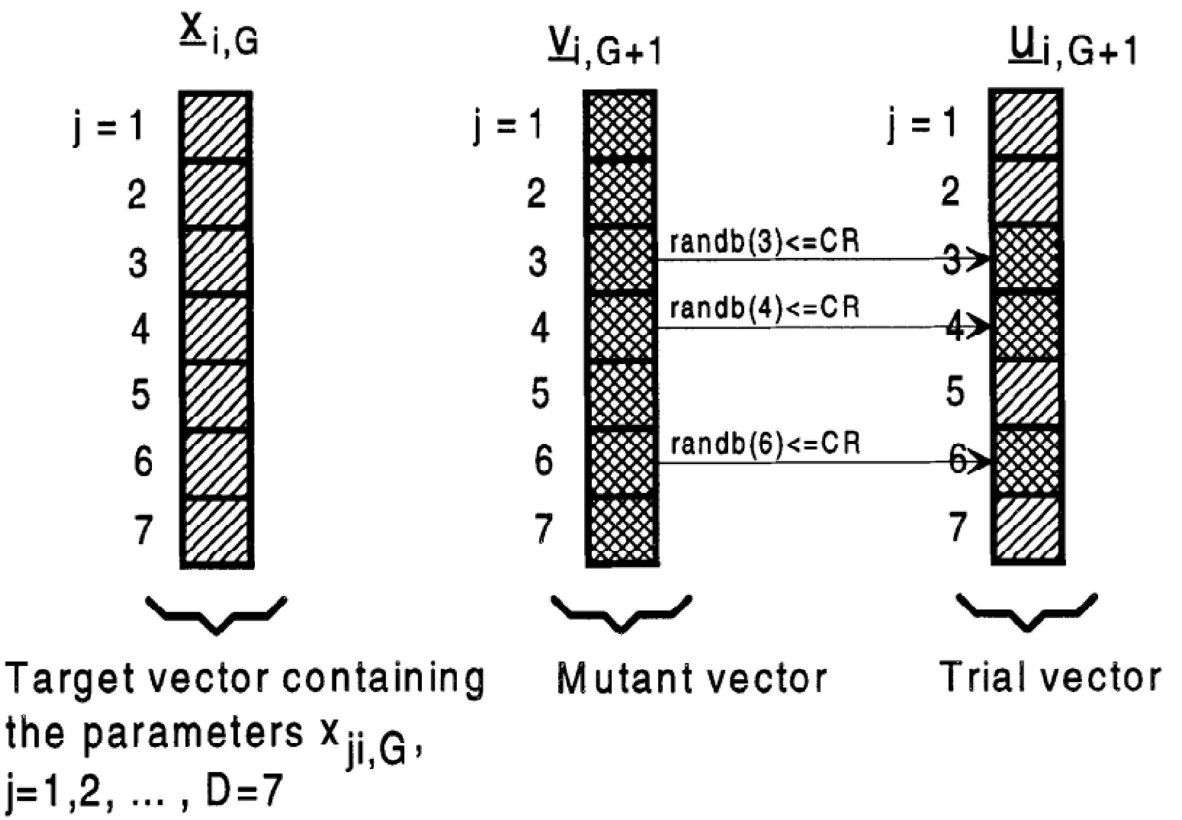


*Figure 2.* Illustration of the crossover process for $D = 7$ parameters.

*Image source: Storn, R., & Price, K.V. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization, 11, 341-359.*

# Selection in DE

- Note that DE "selects" individuals in a way quite different from what we've seen so far in Evolutionary-based metaheuristics method
- DE uses the greedy criterion in the selection process (survival selection) that only accept higher quality solution
  - Trial vector can only replace the target vector if it is more fitter
- Question: What is the gain and loss of this selection strategy?

# Exploration and Exploitation in DE

- **IF** the population is spread out; ($\vec{b}$) and ($\vec{c}$) are likely to be far from one another
  - It makes a "large" mutant vector
- **ELSE**; ($\vec{b}$) and ($\vec{c}$) are likely to be close from one another
  - It makes a "small" mutant vector
- This way, if the population is spread throughout the space, mutations will be much bigger than when the algorithm has later converged on fit regions of the space.

# Takeaway

- DE has the automatic shifting between exploration and exploitation in generating the mutant vector based on the spread of the solutions in the population
- Beside the common controlling parameters, such as the population size and number of iterations, DE has two other important parameters
  - Mutation Rate ($\alpha$) > to control how explorative the mutant vector
  - Crossover Rate ($CR$) > to control how frequent the crossover process
- To run DE, the population size is at least 4
- DE utilizes the spread of the solutions in the population. Thus, larger population size could benefit DE more than it in lower population size