# Bio-Inspired Algorithms for Intelligent Systems

Miguel Leon

# Optimization problems

- In engineering, there are problems that a computer cannot solve in a feasible time as the problems increase.
- This types of problems are called NP-complete. As the problems growth, the computer will need more time to find the optimal solution. And this solution is unknown.
- How much time do you think a simple problem can take?

# Example Problem

- A shipping company wants to send a truck with items that clients bought.
- Each item i has a weight $w_i$and a priority value $p_i$. The truck has a maximum weight of W. The company want to send a truck which maximizes the summation of all the priorities.

- How much time do you think that a computer needs in order to find the best solution?

# Relation between the size of the problem and time used by the computer

| Number of Items | Number of Combinations | Time expend by the computer |
|---|---|---|
| 10 | 1,024 | 0.001 second |
| 20 | 1,048,576 | 0.03 seconds |
| 30 | 1,073,741,824 | 18 minutes |
| 40 | 1,099,511,627,776 | 13 hours |
| 50 | 1,125,899,906,842,624 | 36 years |

Assuming that a computer can calculate 1,000,000 of different combinations in a second.

# Bio-inspired Optimization algorithms

- Ant Colony Optimization
- Artificial bee colony algorithm
- Bat Algorithm
- Differential Evolution
- Firefly Algorithm
- Genetic Algorithm
- Particle Swarm Optimization
- …

# EVOLUTIONARY COMPUTATION (EC)

- EC: simulate natural selection in a computer program to improve system performance automatically
- Like natural selection, EC aims to facilitate stochastic search in problem space
- EC conducts randomized, parallel beam search. It becomes more popular with advancement of computer hardware.

# Family of Evolutionary Computation

- **Genetic algorithms**
- **Genetic programming**
- **Differential Evolution**
- Memetic algorithm

# AGENDA

- Genetic algorithm
  - What is it?
  - Types
  - Important parts
- Differential Evolution
- Genetic Programing
- Ant Colony Optimization
- Particle Swarm Optimization

# Genetic Algorithms

# What is a Genetic Algorithm (GA)?

**The genetic algorithms**

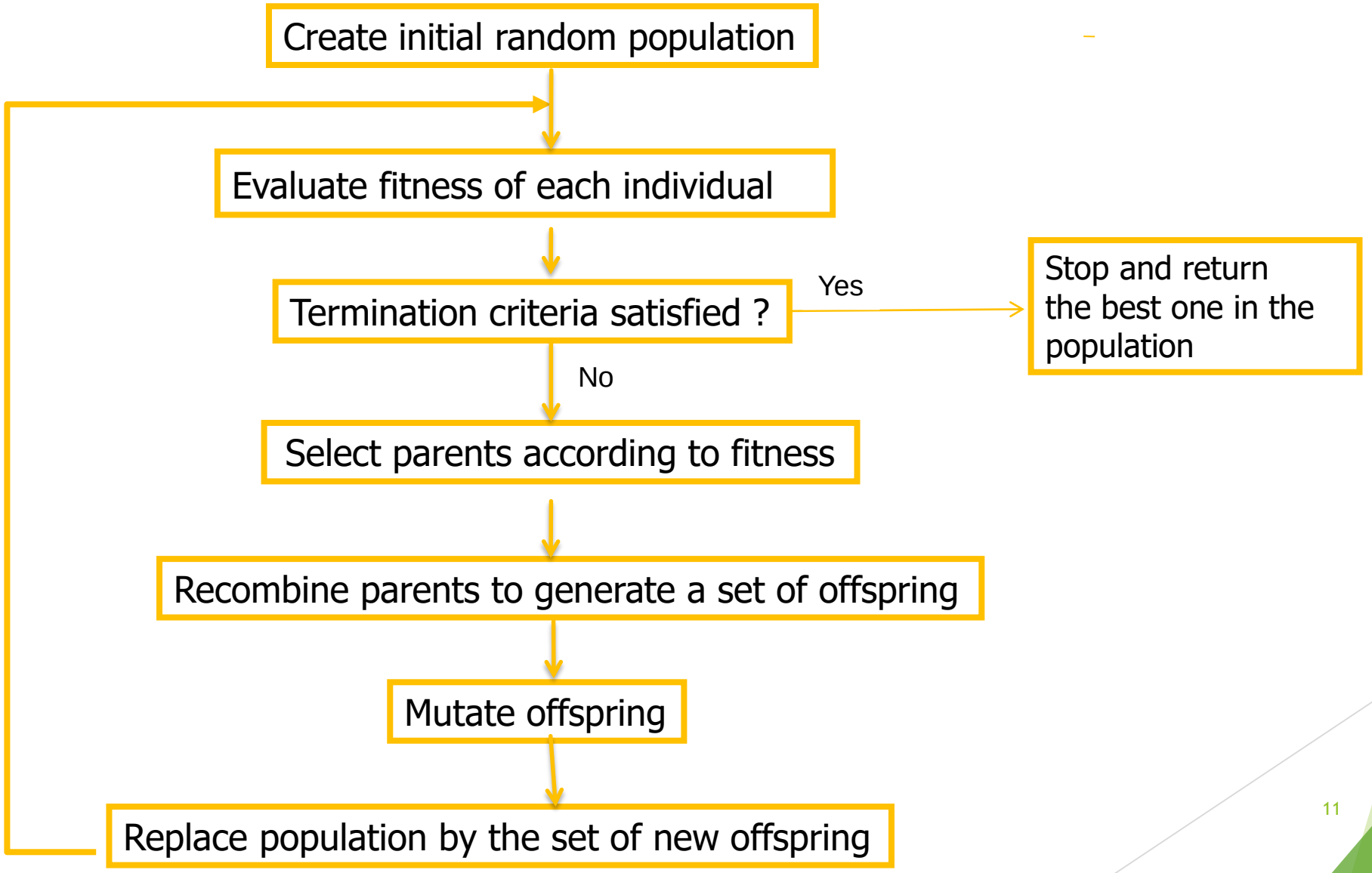
are a class of algorithms for optimization, search and learning, which are inspired by

**Natural Evolution**

**and**

**Genetic Evolution**

# Flow Chart of Traditional Genetic Algorithm



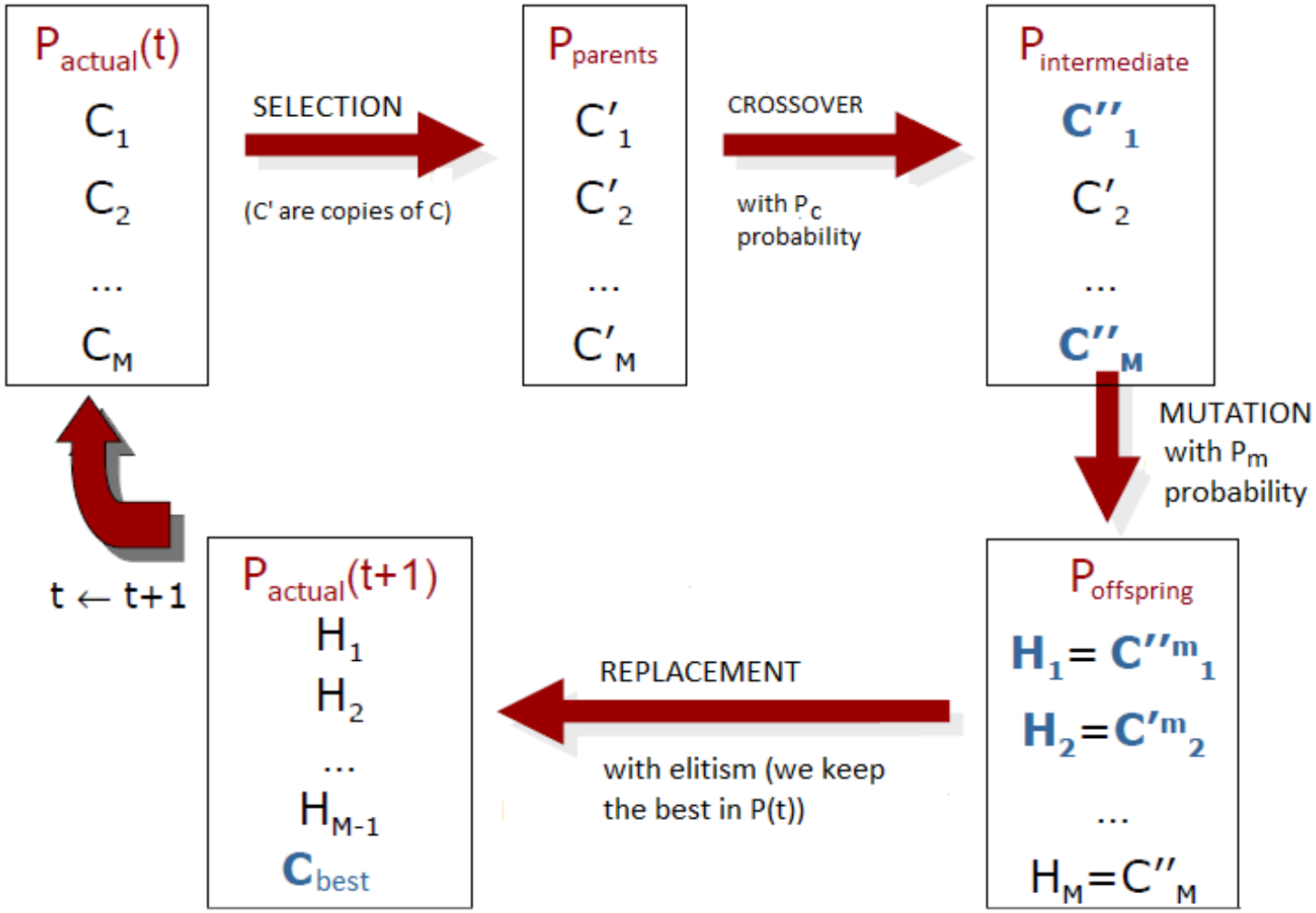
# Two models: Gerenational and stationary

- **Generational model:** A completely new population of new individuals is created in every iteration.

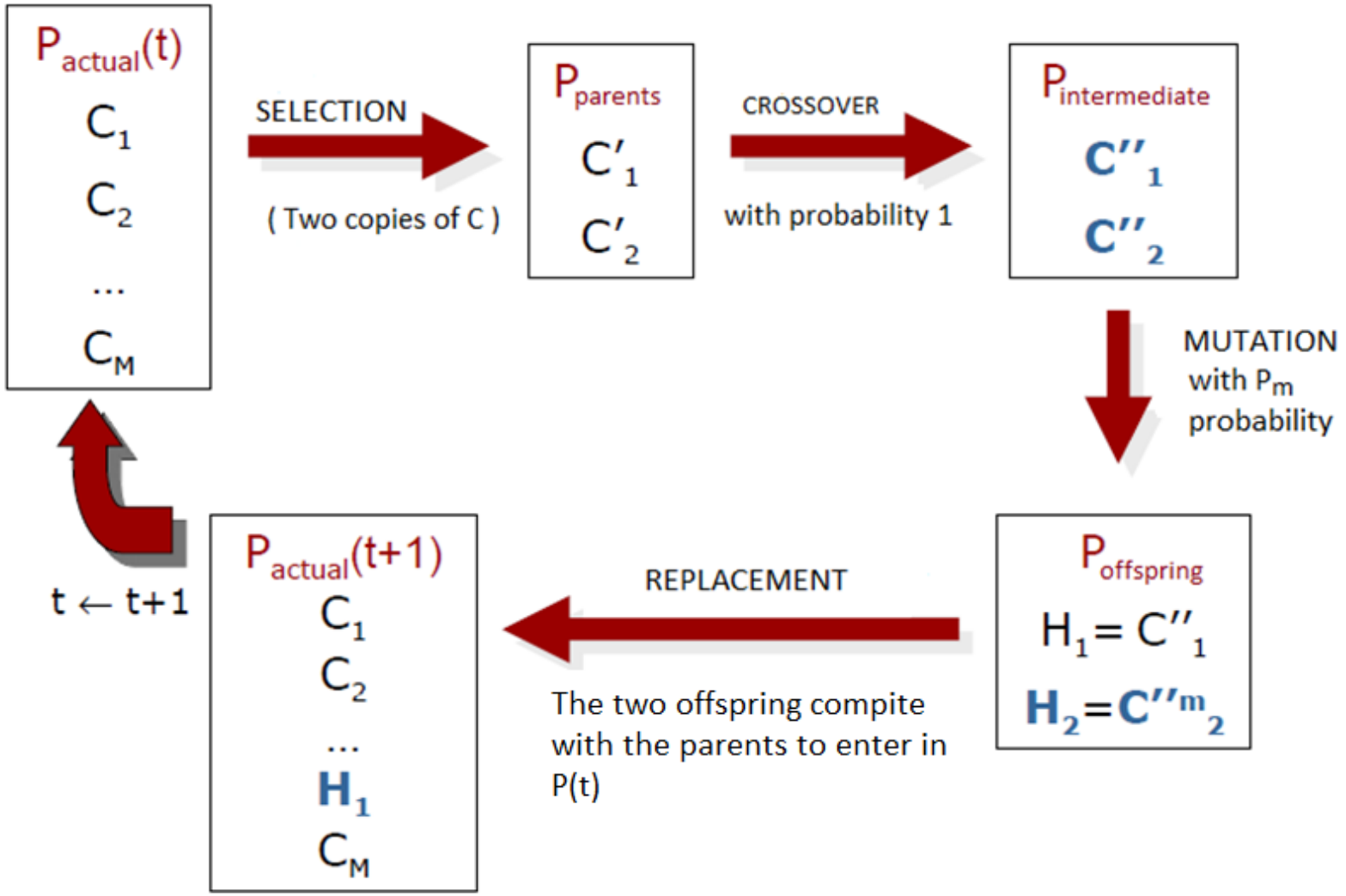
  **The new population replace the old one.**

- **Stationary model:** Two parents are chosen and the genetic operators (crossover and mutation) are applied to produce offspring. **Only the best individuals from the parents and offspring survive in the next generation.**

# Generational model



# Stationary model



# How to build a genetic algorithm?

- Design a representation
- Decide how to initialize a population
- Design a way to evaluate an individual (fitness function)
- Decide how to select the parents
- Design a crossover operator
- Design a mutation operator
- Decide how to replace the individuals
- Decide when to stop

# Representation

There is several kinds of representation. The two more common are:

- Binary representation:

| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

- Real representation:

| 7 | 3 | 6 | 8 | 2 | 4 | 1 | 5 |
|---|---|---|---|---|---|---|---|

# Intizialize

Two ways:

- Randomly created in the search space:
  - Binary representation: 0 or 1 with probability 0.5
  - Real representation: Uniform inside the range.

- Choose the initial population to contain solution(s) from a previous search.

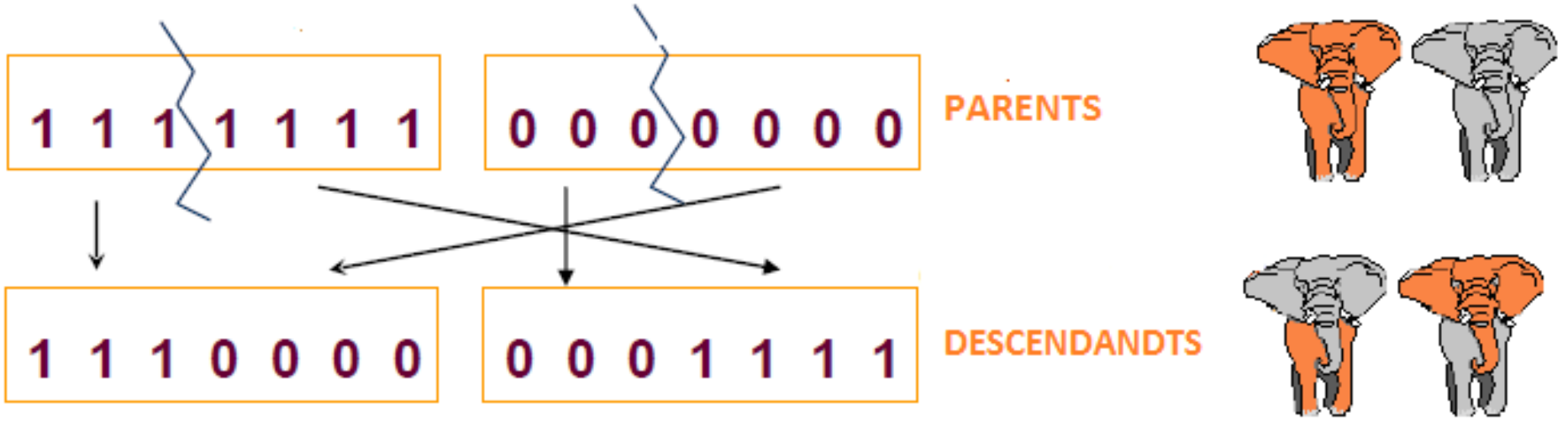
# Selection strategy

- **Tournament Selection (TS):** Choose the best from the k randomly selected individuals from the population. K is the size of the tournament.
- **Linear Order(LS):** The population is ordered according the fitness value and then a probability is given according to the order.
- **(NAM):** The first parent is chosen randomly. The other is chosen according to the distance with the first one. So the most different individual will be chosen.
- **Roulette Selection:** A probability is assigned to an individual, proportional to the fitness of this individual.

$$P(S_i) = \frac{Fit(S_i)}{\sum_j Fit(S_j)}$$

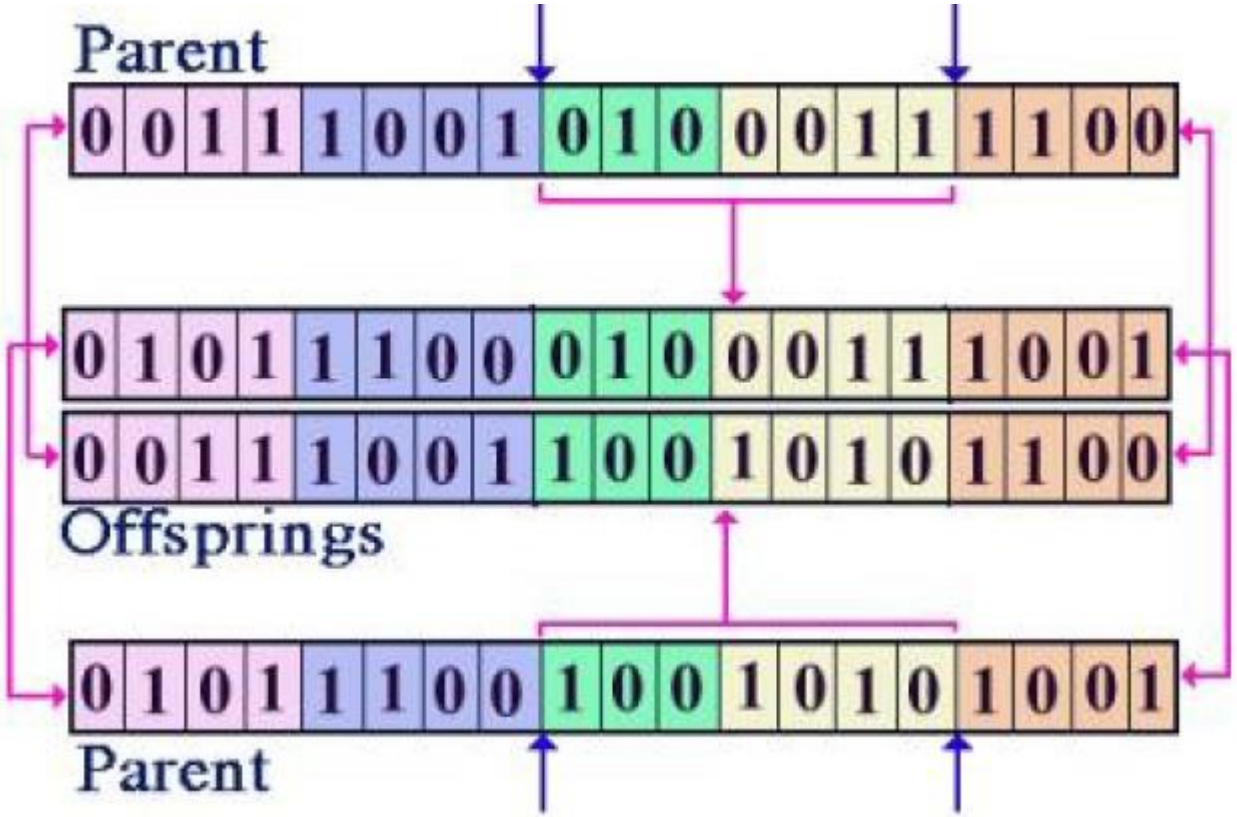
# Crossover (1)

Crossover operator with binary representation (example 1):



# Crossover (2)

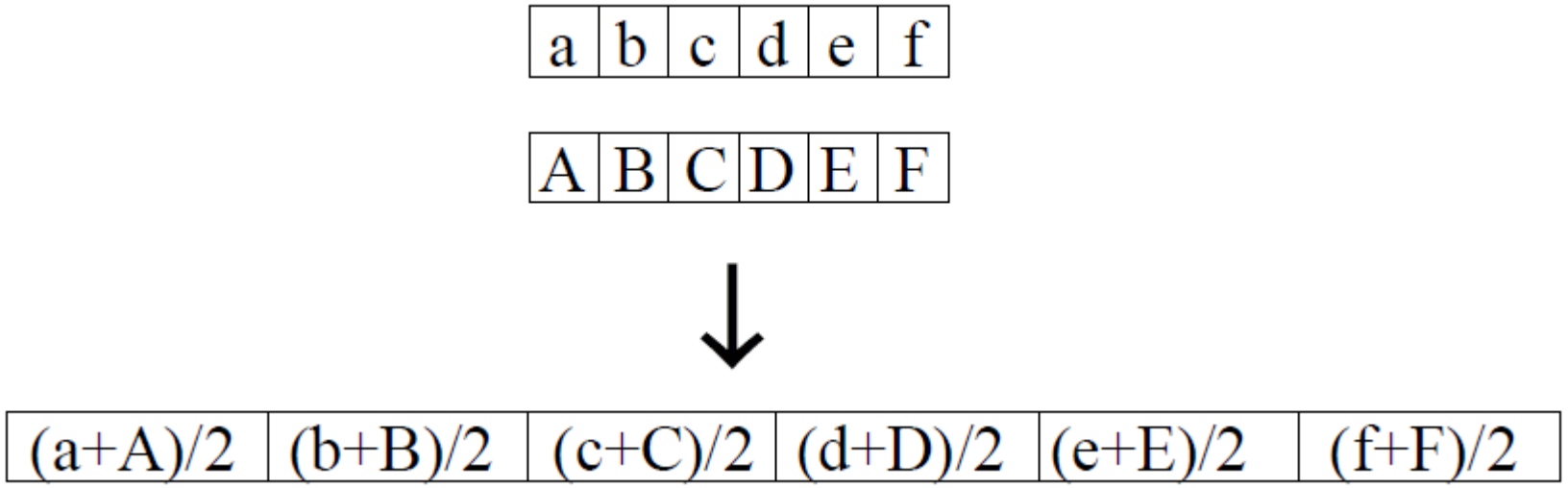
Crossover operator with binary representation (example 2):



# Crossover (3)

Crossover operator with a real representation (Example 1).

**Arithmetic Recombination.**



# Crossover (4)

Crossover operator with a real representation (Example 2).

**BLX-α.**

- Given 2 individuals

$$C_1 = (c_{11},\ldots, c_{1n}) \text{ and } C_2 = (c_{21},\ldots, c_{2n}) ,$$

- BLX- α creates two descendants

$$H_k = (h_{k1},\ldots, h_{ki},\ldots, h_{kn}) , k = 1,2$$

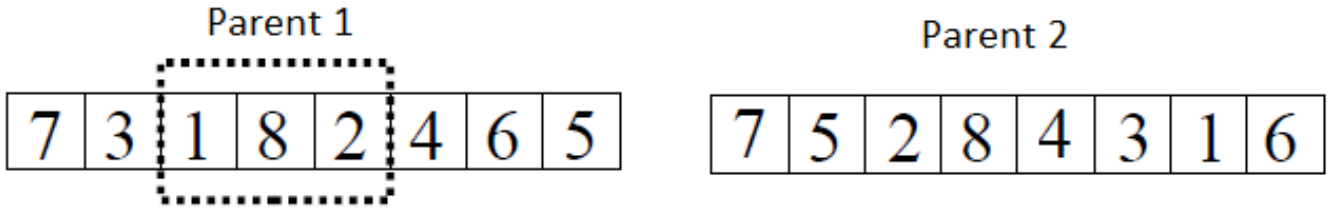
- where $h_{ki}$ is created randomly in the interval:

$$[C_{min} - I\cdot\alpha, C_{max} + I\cdot\alpha]$$

  - $C_{max} = \max \{c_{1i} , c_{2i}\}$
  - $C_{min} = \min \{c_{1i} , c_{2i}\}$
  - $I = C_{max} - C_{min} , \alpha \in [0,1]$

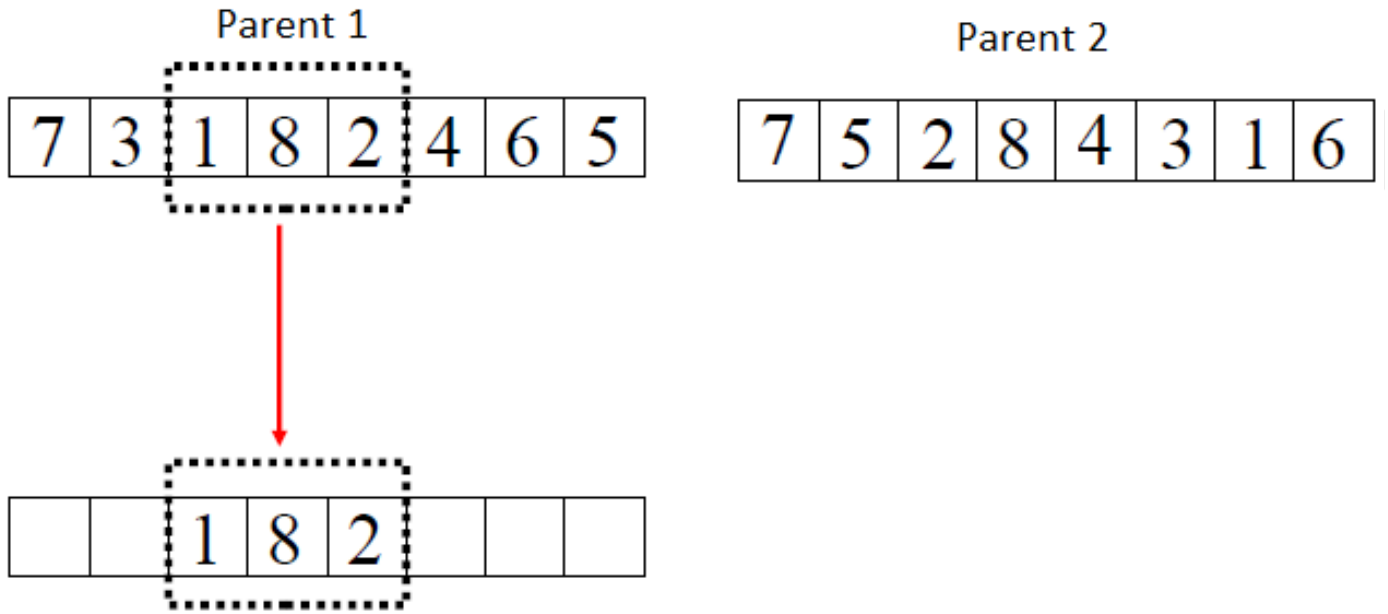
# Crossover (5)

- Crossover operator with a order representation (Example 1).



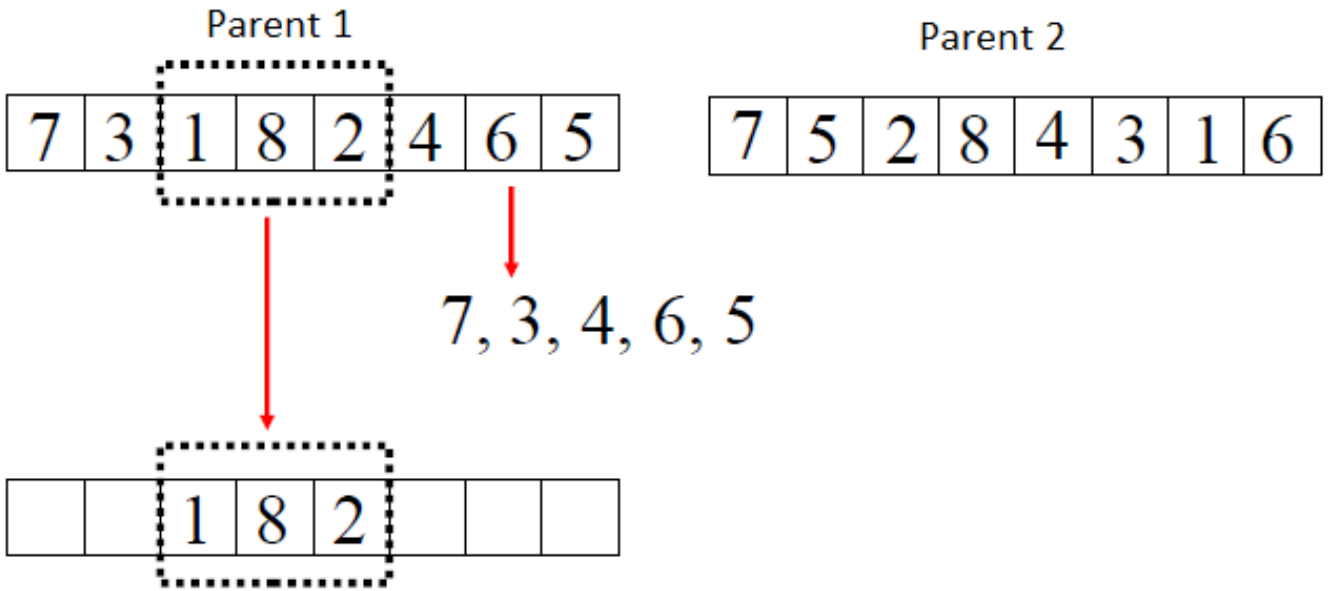
# Crossover (5)

- Crossover operator with a order representation (Example 1).



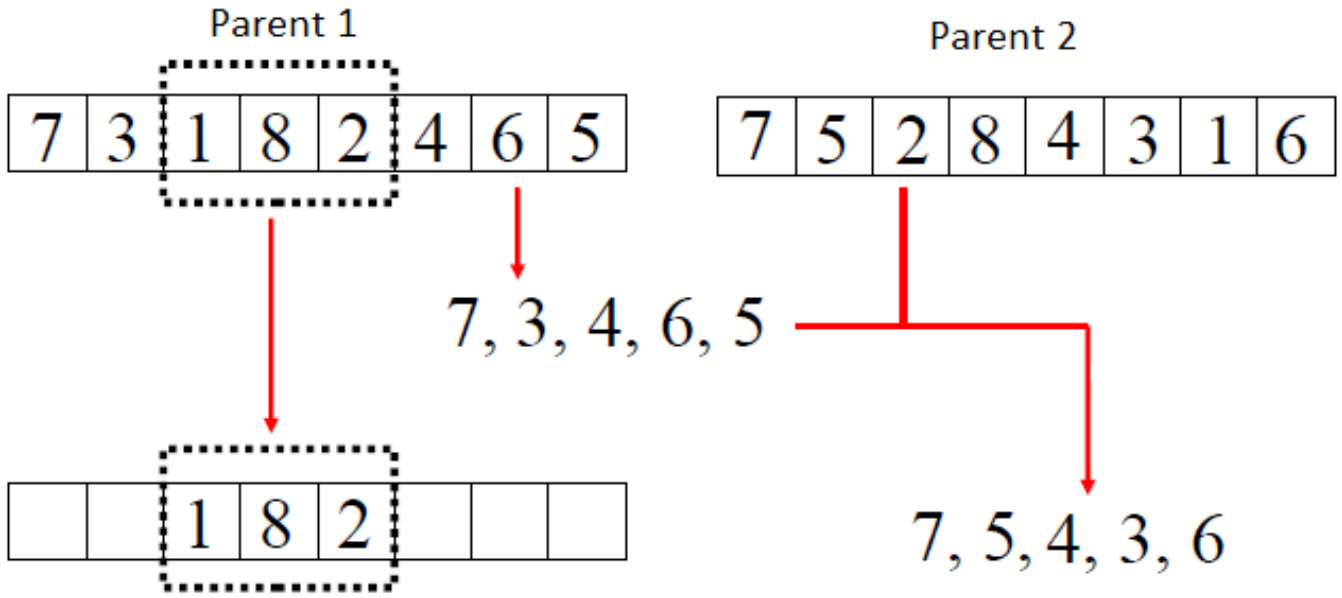
# Crossover (5)

- Crossover operator with a order representation (Example 1).



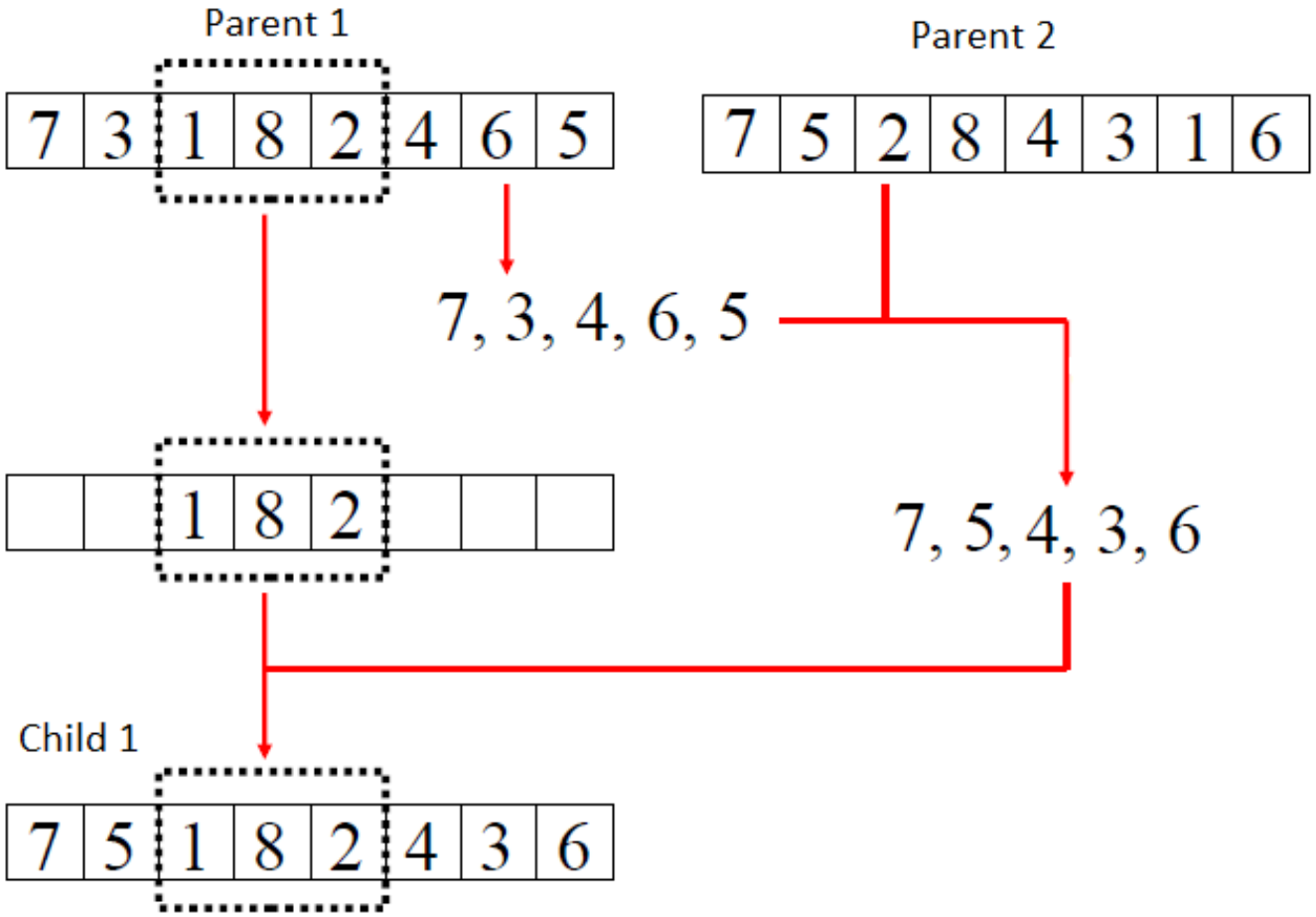
# Crossover (5)

- Crossover operator with a order representation (Example 1).



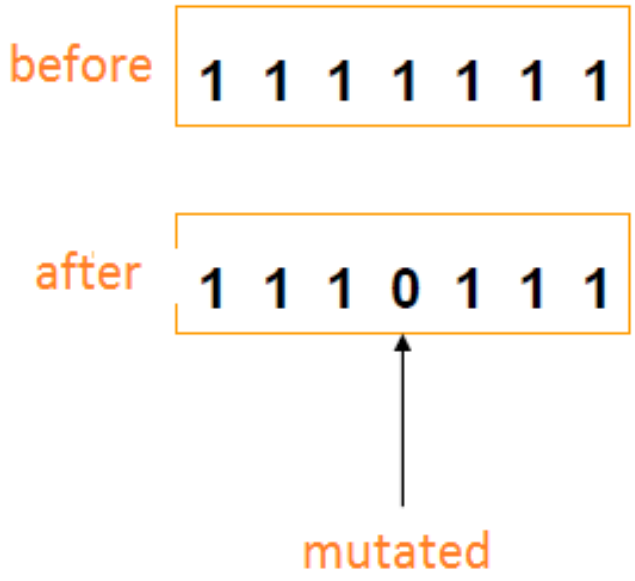
# Crossover (5)

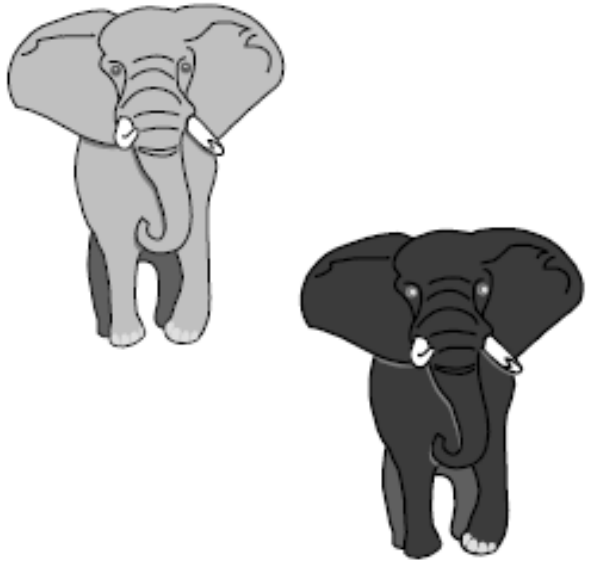
- Crossover operator with a order representation (Example 1).



# Mutation (binary representation)

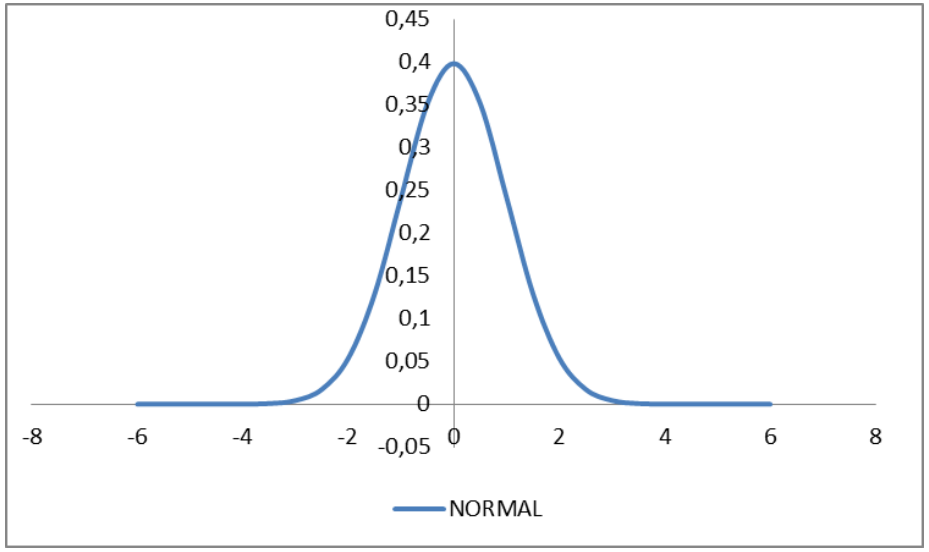
Mutation operator with binary representation:



# Mutation (real representation)

- Given an offspring string $X=[x_1, x_2, \ldots., x_n]$, mutation can be done by adding to each number in the string a random disturbance $\boldsymbol{u_i}$, which is subject to a normal density function $N(0, \delta)$.




- Consequently, we will have mutated offspring as:

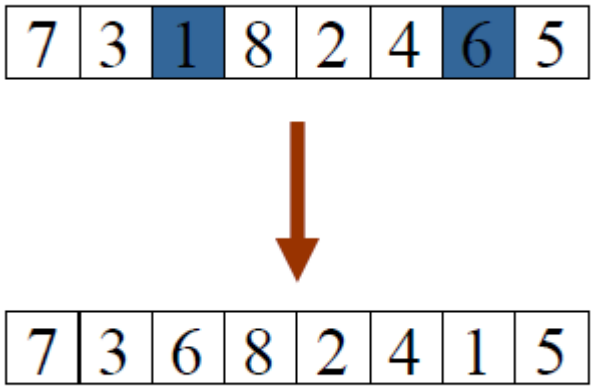
$$X'=[x'_1, x_2, \ldots., x'_n]$$

with $$x'_i=x_i+u_i$$

- For example, in matlab we have the function sqrt(δ) * randn(1).

# Mutation (order representation)

- Select two random positions and change the elements.



# Replacement (generational model)

- The offspring replace the entire population.
- We can decide to keep the best individual: **ELITISM**

  (It is recommended to use elitism)
- It is possible to have high grade of elitism, keeping N best individuals in the population.

# Replacement (stationary model)

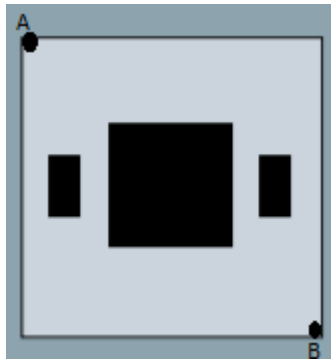
- **Replace the worst in the population (RW).**
- **Restricted tournament (RTS):** The offspring replaces the most similar individual among w (w=3,4,…) individuals previously selected.
- **Worst between similar (WAMS):** We select a group of individuals most similar to the offspring individual. The offspring replaces the worst individual in the group.

# Stop criterion

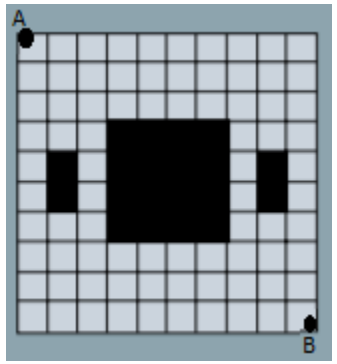
- If we find the optimum!!
- Maximum number of evaluations.
- Maximum number of iterations (generations)
- Limit of the user:
  - Time
  - After some iterations without improvement

# Navigation problem

- We have the following map:




- How can we go from A to B using GA?
  - First we have to divide the map in squares (this is only a possible option):

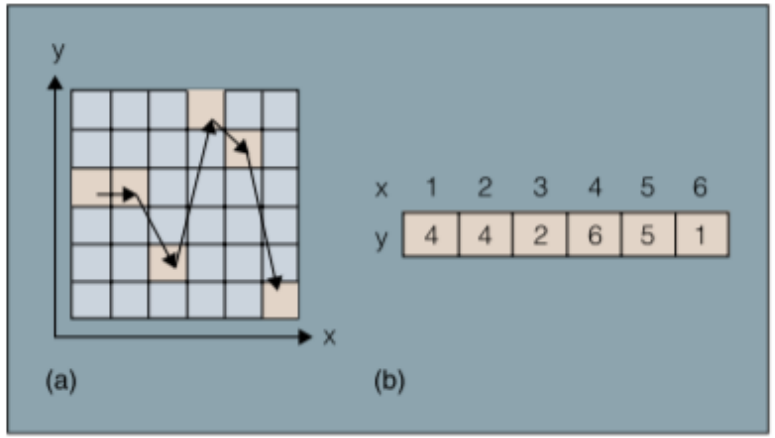


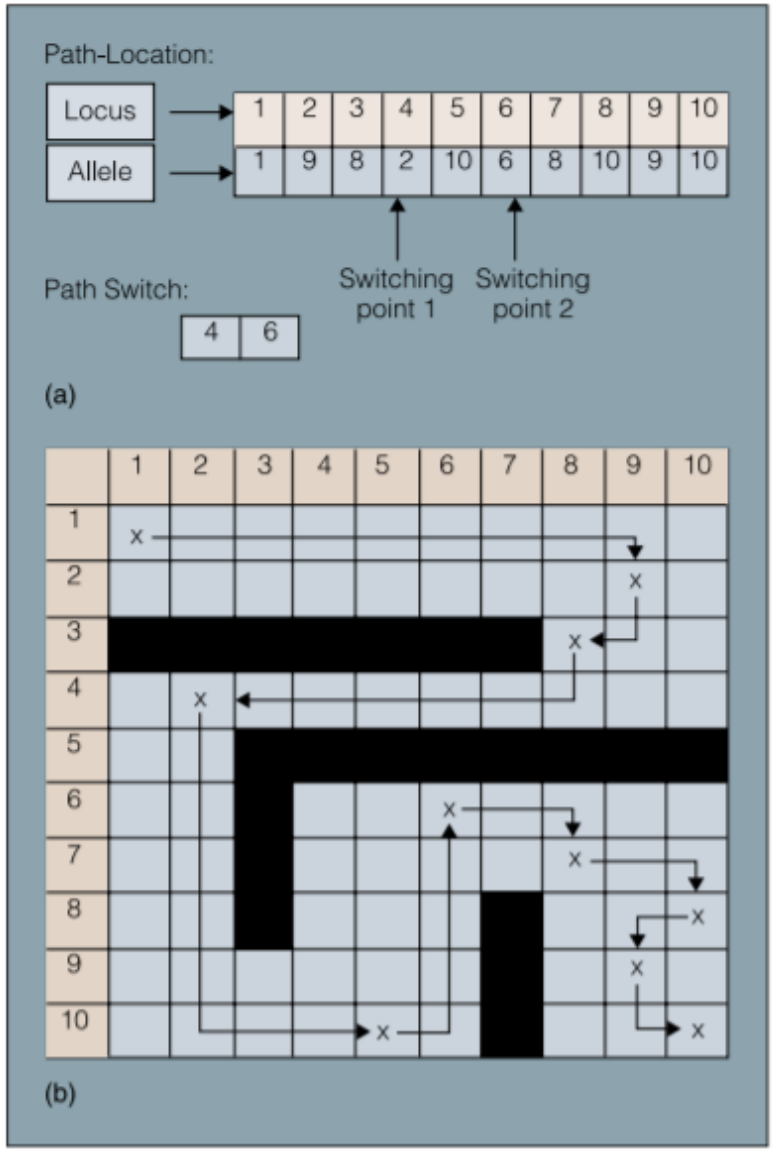

- Source: T. Manikas, K. Ashenayi, R. Wainwright. Genetic Algorithms for autonomous robot navigation. IEEE instrumentation and measurements, 10, 6.

# Navigation problem (2)

- When using GA what are the questions you have to answer?
  - How do you represent a possible solution to the problem?
  - What are the restrictions to the problem?
  - How do we evaluate the performance of a solution?
  - How do you perform selection, crossover ....?
  - When to stop the search?

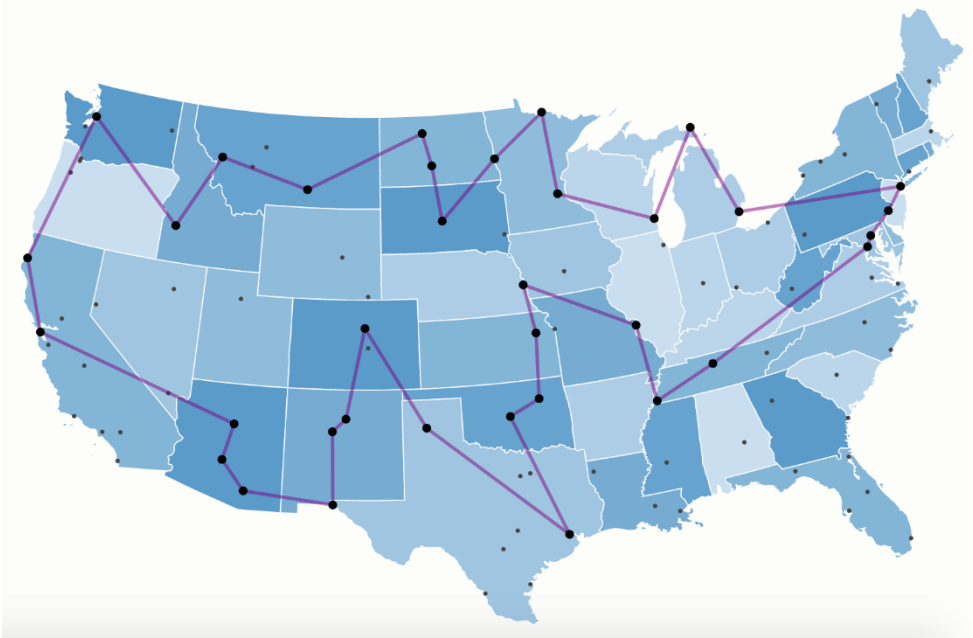
# Navigation problem (3)



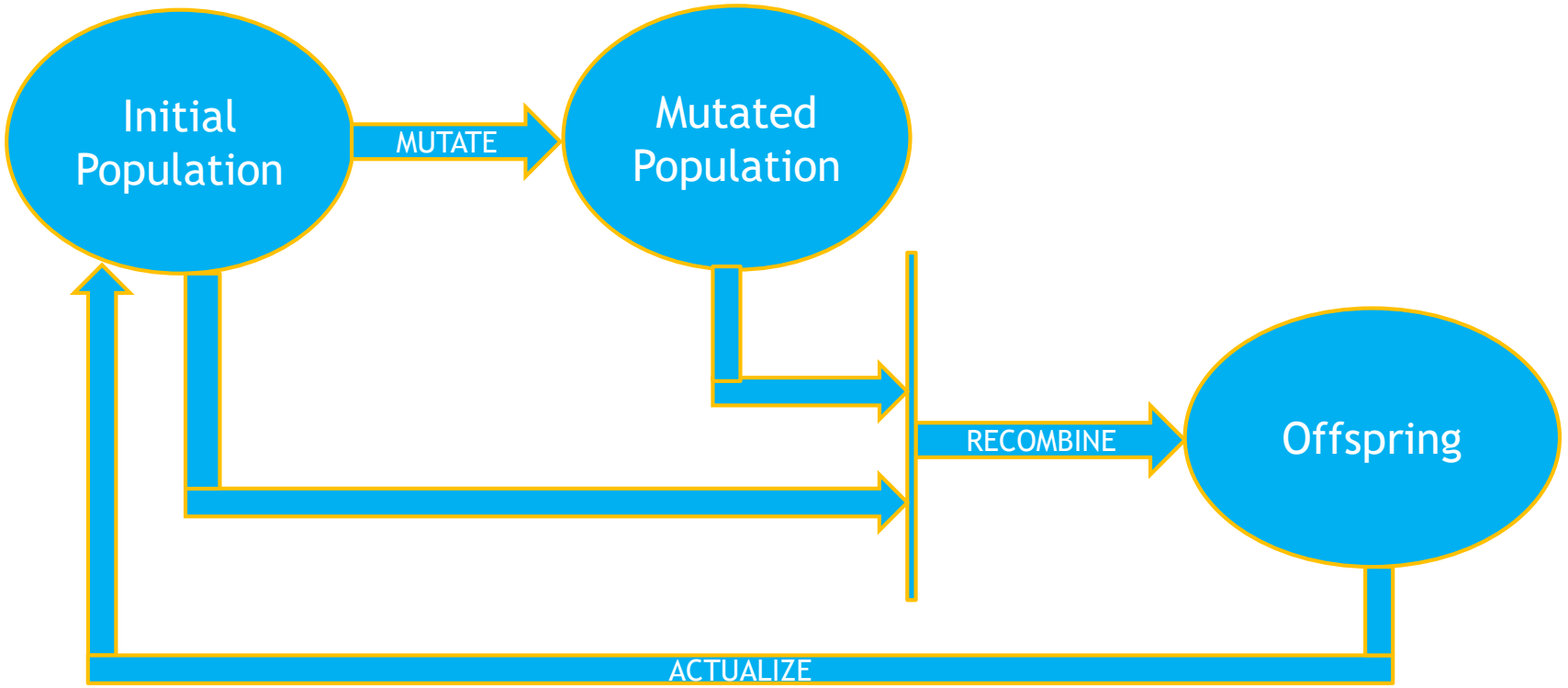
# Travelling Salesman Problem

- We have a number of cities in a map. We have to visit all of them onlly once. Which is the path with the minimum distance?

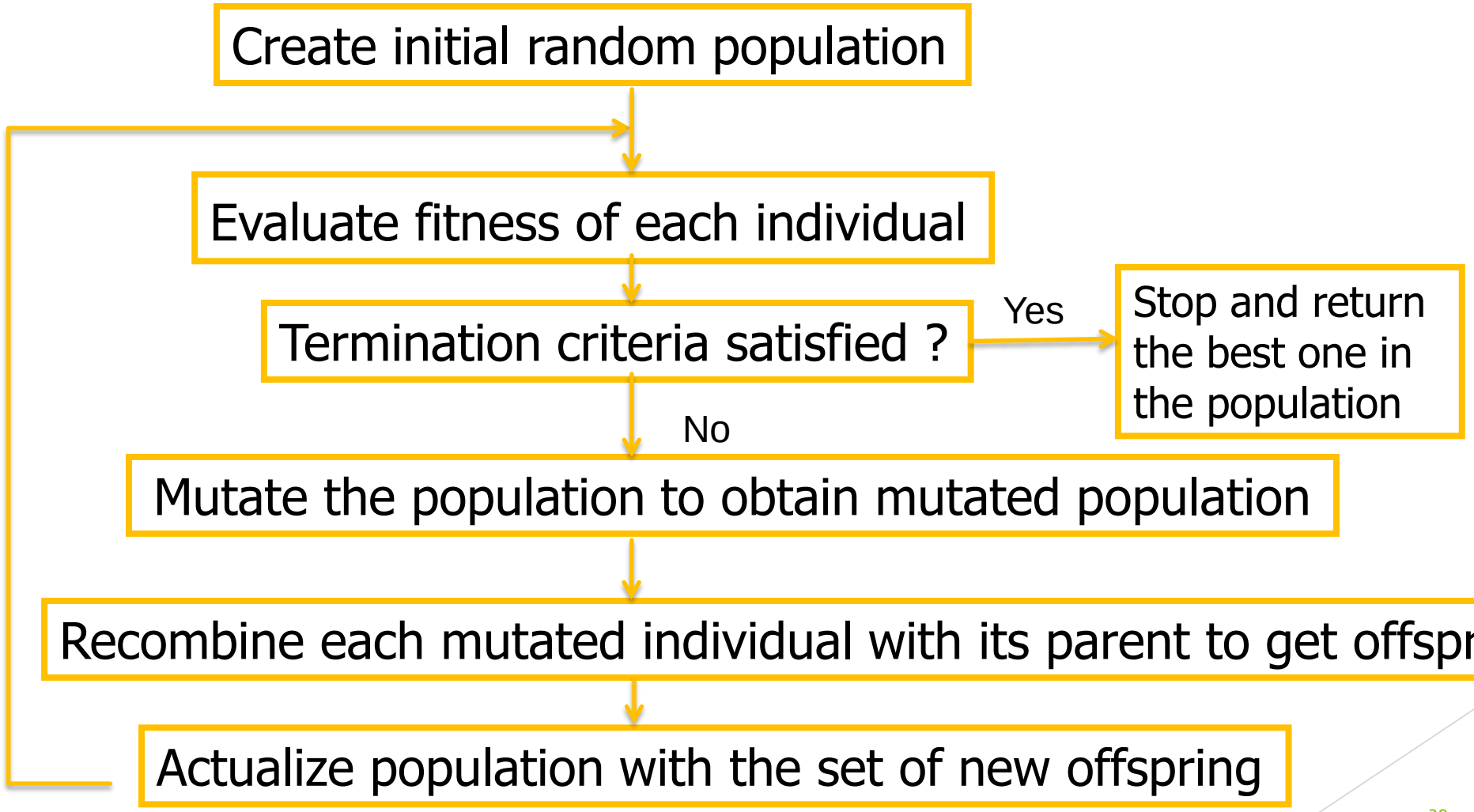


- Now, Imaging that those cities are objects in a room and your vehicle has to find the order in which each objects is picked. How could you apply GA?

# Differential Evolution (DE)

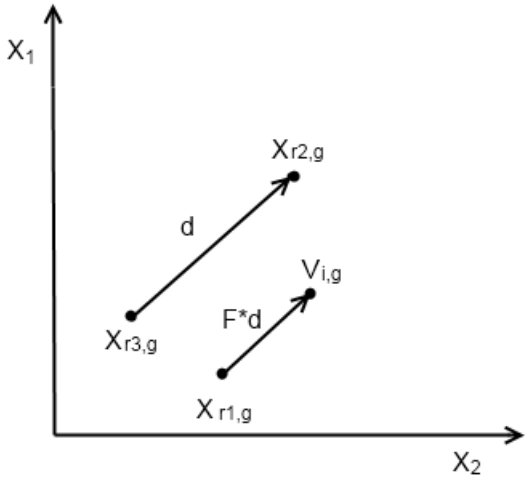


# Flow chart of Differential Evolution



# DE - Mutation

- The most common mutation strategy is Random Mutation Strategy, it is represented by DE/rand/1
- Equation: $V_{i,g} = X_{r1} + F * (X_{r2} - X_{r3})$




Vi,g: mutated individual
Xr1,g , Xr2,g and Xr3,g: individual from the population randomly chosen
F: Scaling factor, $F \in [0,2]$

# DE - Recombination

- The most common recombination strategy is binomial, it is represented by bin.
- Equation:

$$Ti,g[\mathrm{j}] = \begin{cases} Vi,g[j] \; if \; randValue < \mathrm{Pr} \; or \; j = jrand \\ Xi,g[j] \qquad\qquad\qquad\qquad otherwise \end{cases}$$

Ti,g: Trial vector (one offspring)
Vi,g: mutated individual
Xi,g: individual i from the population
Pr: Recombination Probability, Pr ∈ [0,1]
randValue: Random Value between 0 and 1

# DE - Selection

- The last step is to actualize the population to the generation g + 1.
- Equation (searching minimum):

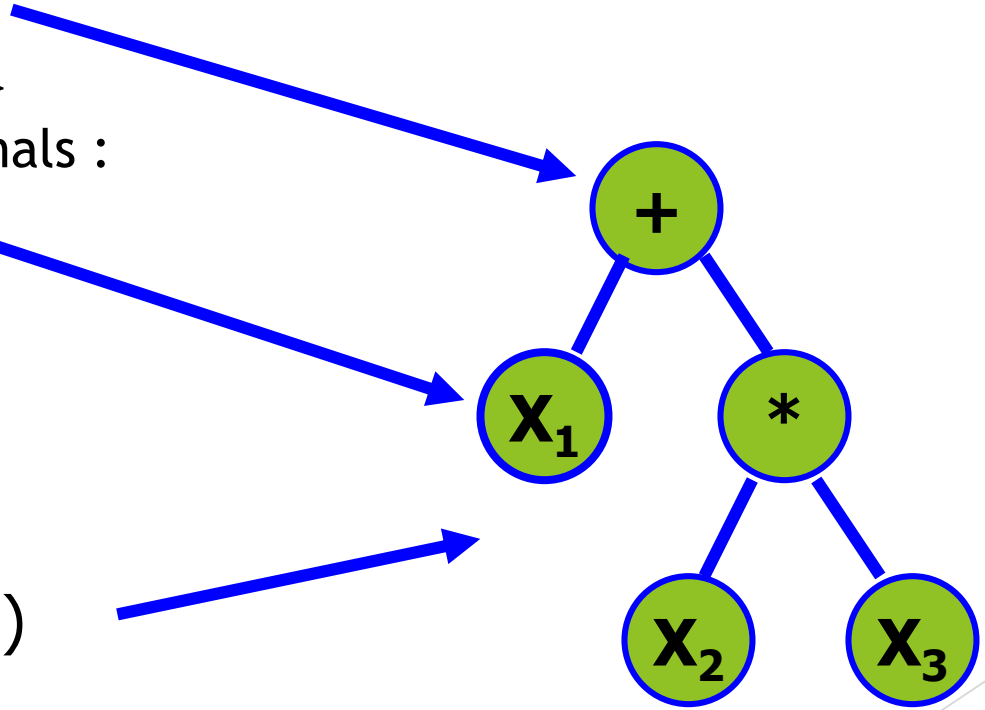
$$Xi, g+1 = \begin{cases} Ti, g \ if \ f(Ti, g) < f(Xi, g) \\ Xi, g \qquad\qquad otherwise \end{cases}$$

Ti,g: Trial vector (one offspring)
Xi,g: Individual i from the population
Xi,g+1: Individual i in the population for the next generation
f(vector): fitness of the vector

# Genetic Programming

- Automatic generation of computer programs by means of natural evolution
- Individuals in population are programs represented as trees
- Tree nodes correspond to functions :
  - arithmetic functions {+,-,*,/}
  - logarithmic functions {sin,exp}
- Leaf nodes correspond to terminals :
  - input variables $\{X_1, X_2, X_3\}$
  - constants {0.1, 0.2, 0.5 }

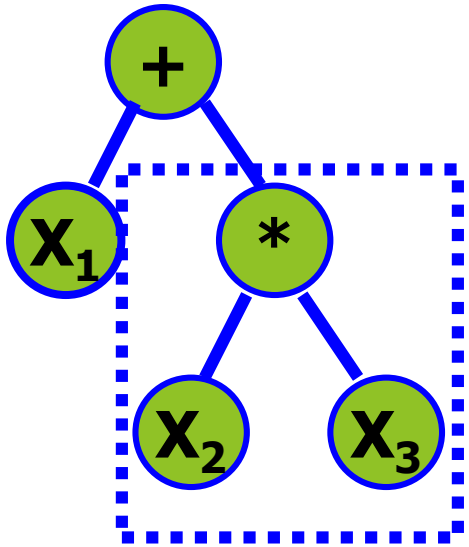
$X_1+(X_2*X_3)$



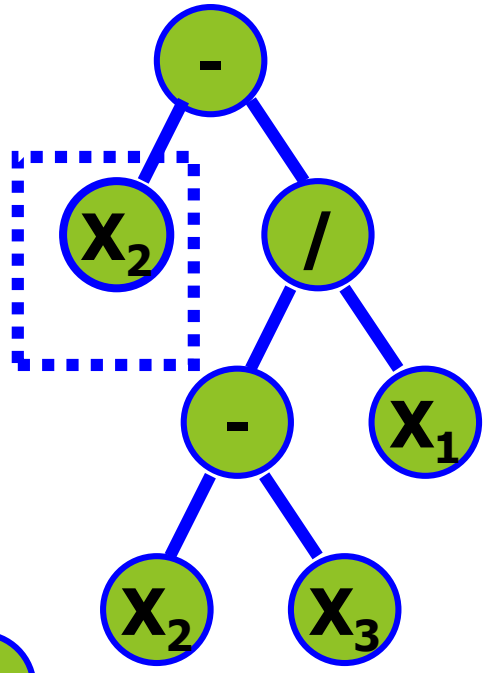
# Genetic Programming : Crossover

Crossover is performed by exchanging randomly chosen parts between parents

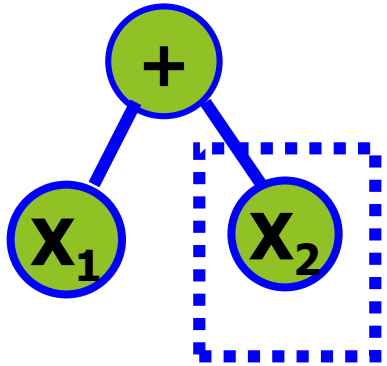
parent A

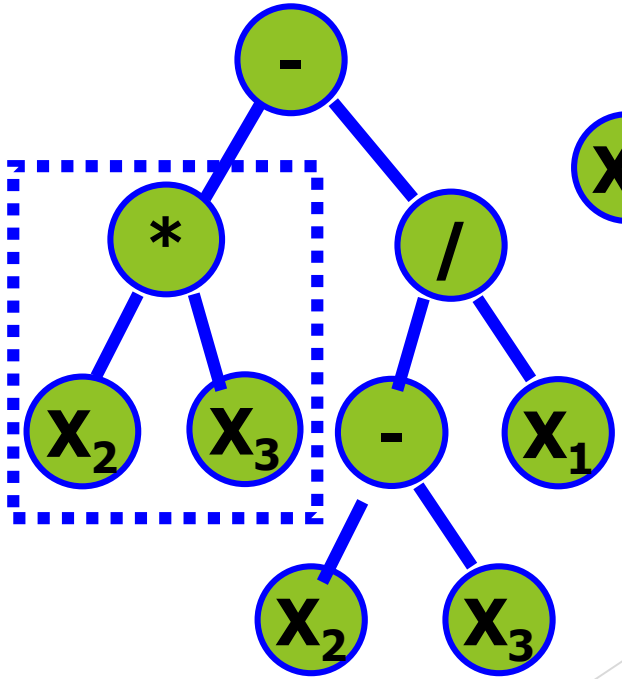



parent B




offspring A

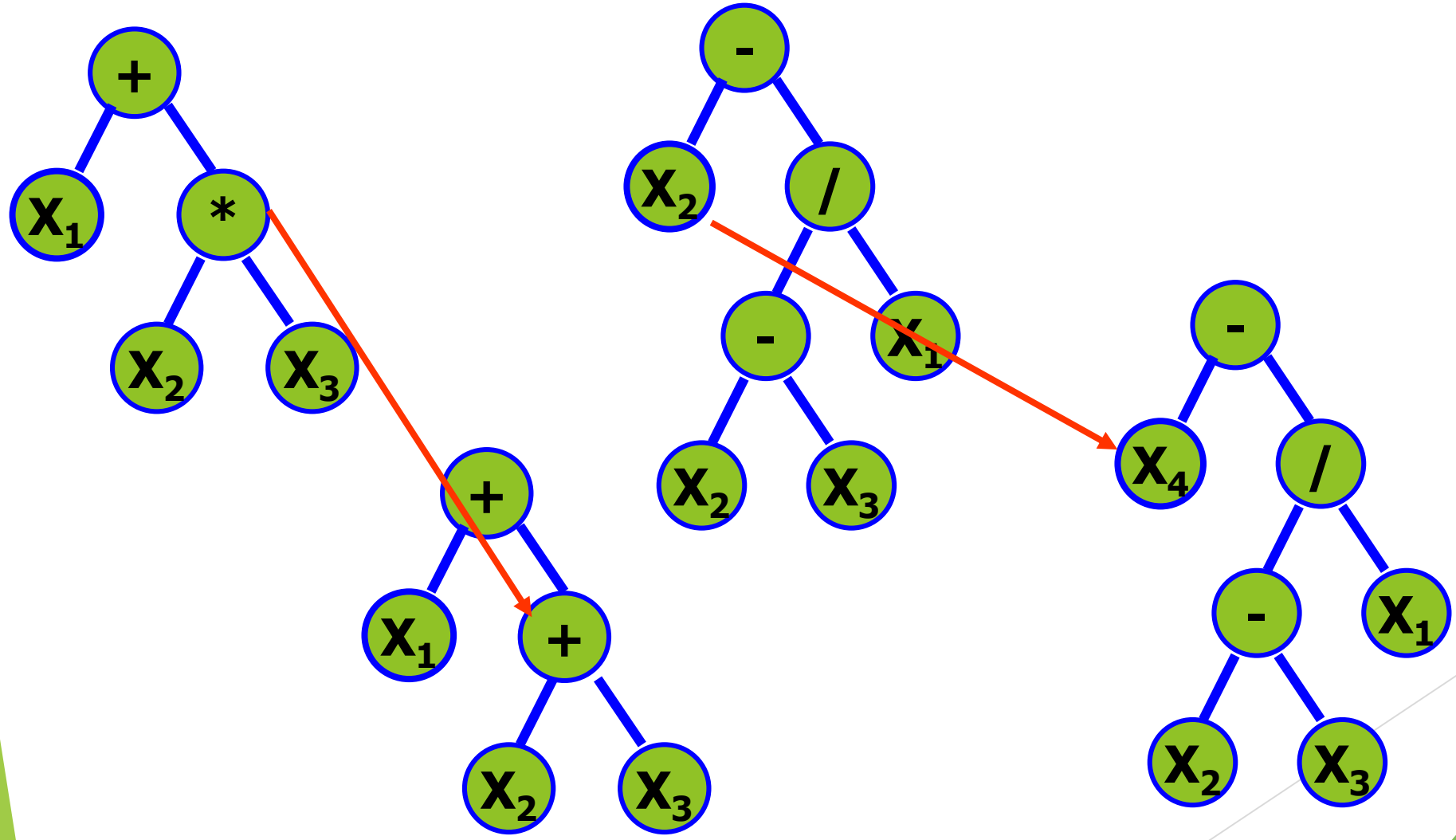



offspring B

# Genetic Programming: Mutation

- A mutation operator can randomly change any function or any terminal in the tree.



# Particle Swarm Optimization

# Particle Swarm Optimization (PSO)

- Particle Swarm Optimization is a population based algorithm inspired by the social flight of the birds and the movement of the fishes.
- PSO was created in 1995 by Russ C. Eberhart and James Kennedy.
- PSO is used in continuous optimization.

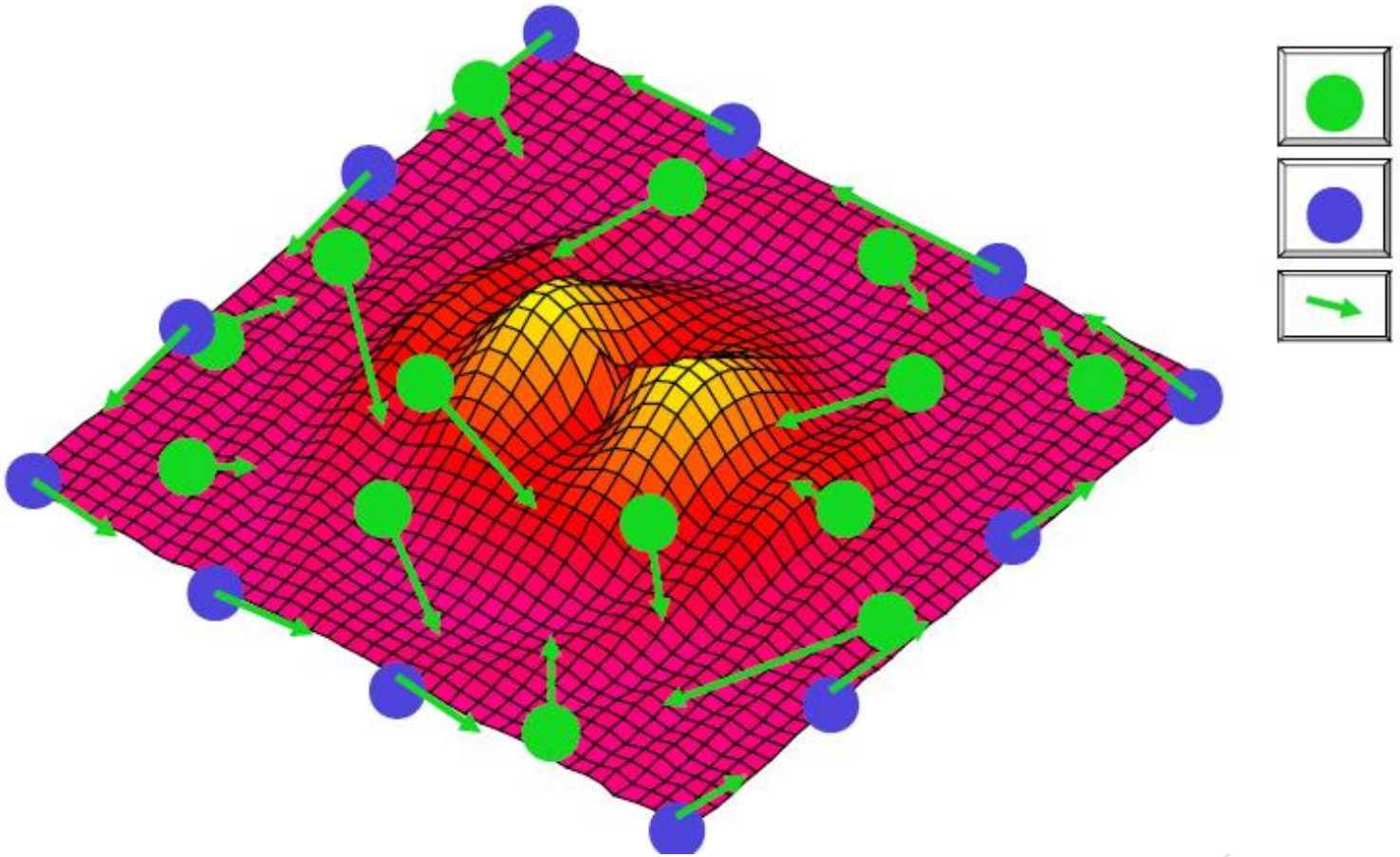
# Basics

- PSO simulate the behavior of the flock of birds.
- Imagine that one of this flock is searching food in an area and there is only one piece of food in that area.
- The birds do not know were the food is but they know the distance to it.
- The most efficient strategy will be to follow the bird that is closer to the food.

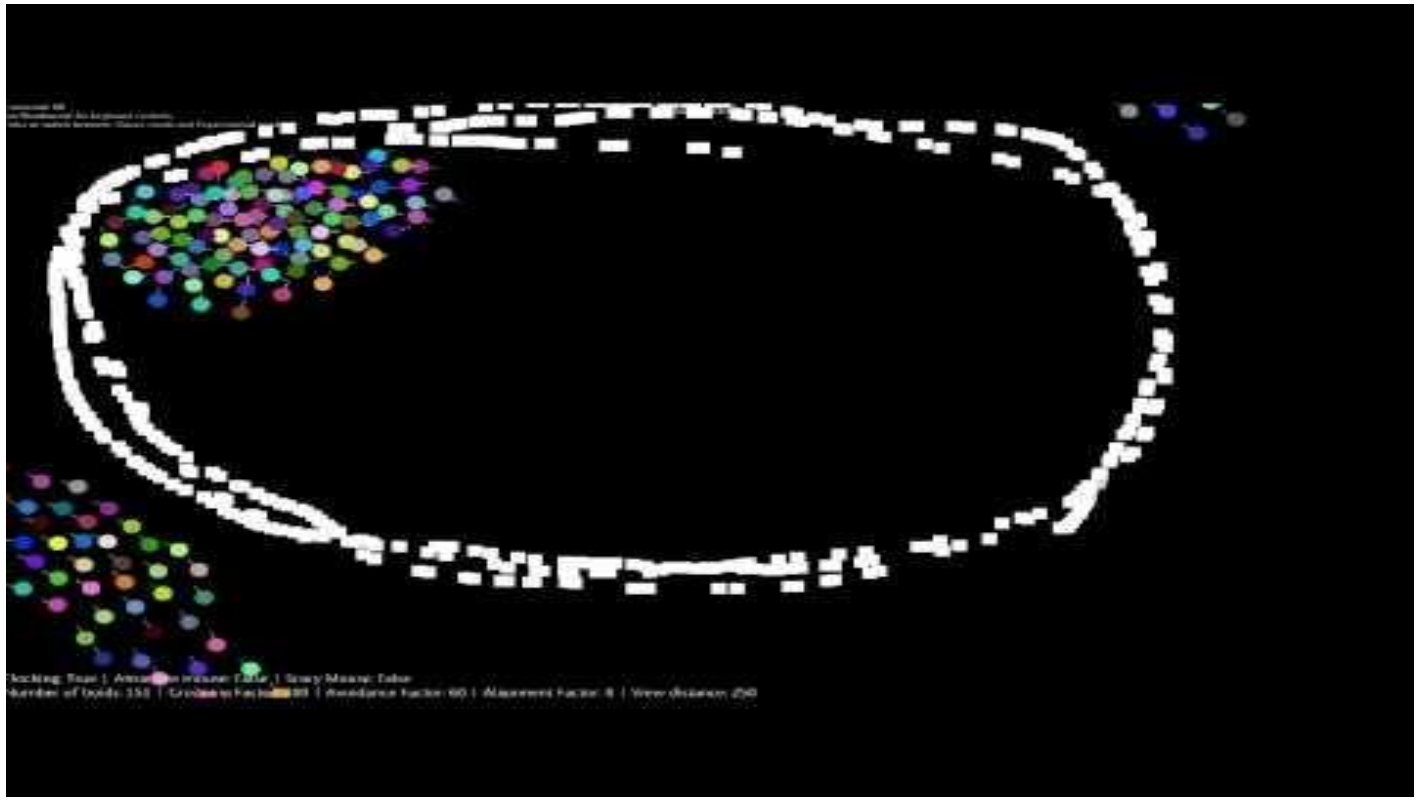
# Basics (2)

- The population will be a multi-agent system where every bird or particle will be an agent.
- Every particle will have a fitness (in this case distance to the food), a position in the space and a velocity vector (the direction that the particle is tacking)

# Initialization of the particles



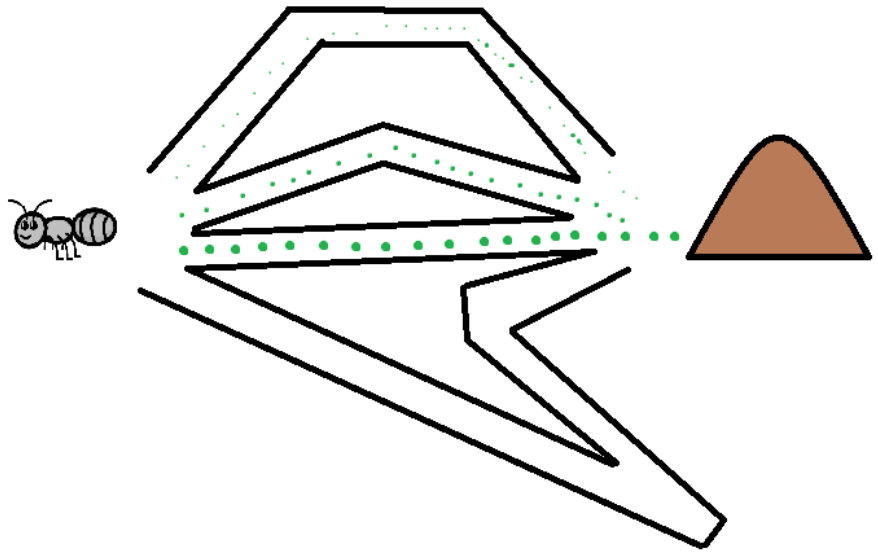
# Multiple robot searching



# Ant colony optimization

# Ant Colony Optimization (ACO)

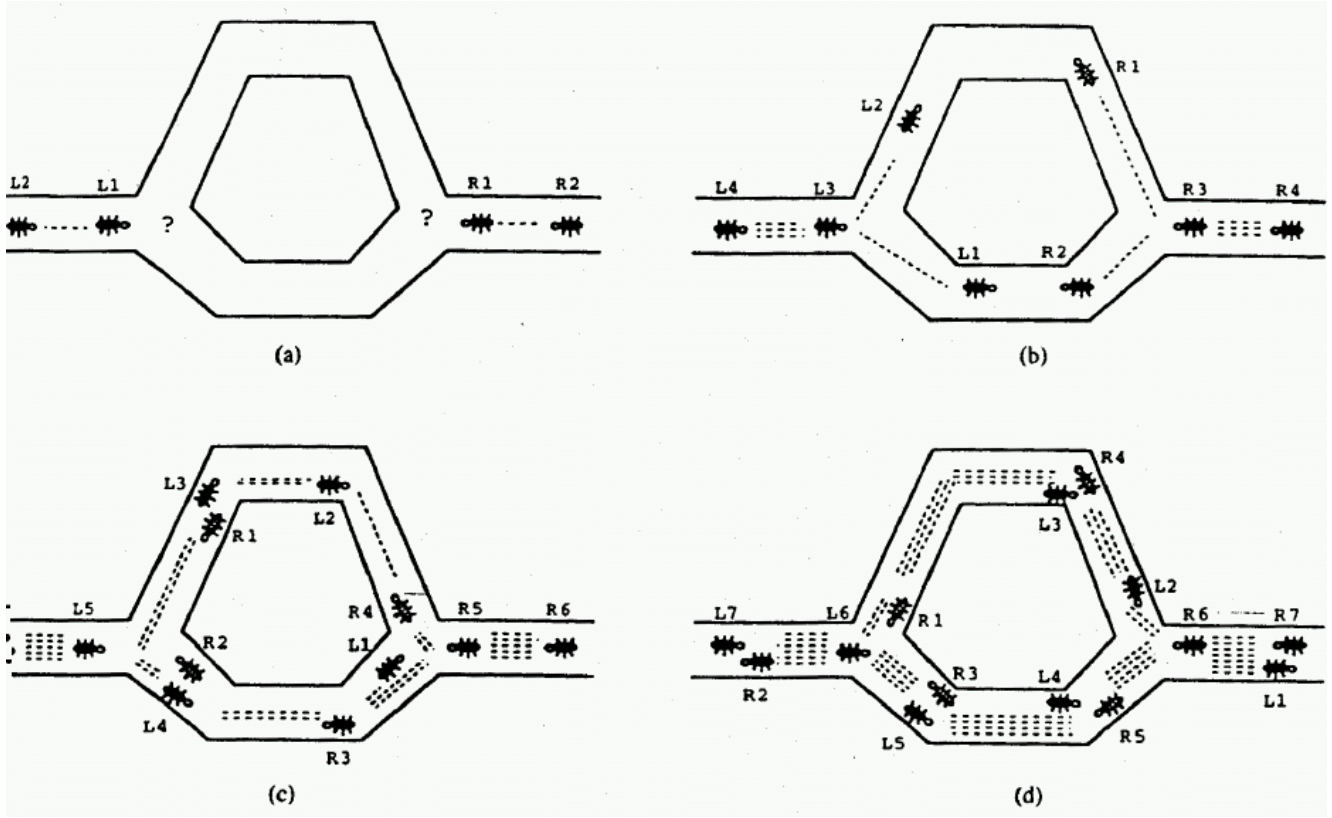
- Ant Colony System was invented by Marco Dorigo in 1992
- This algorithm is based on the behavior of real ants
- It is used to find the shortest path in a graph



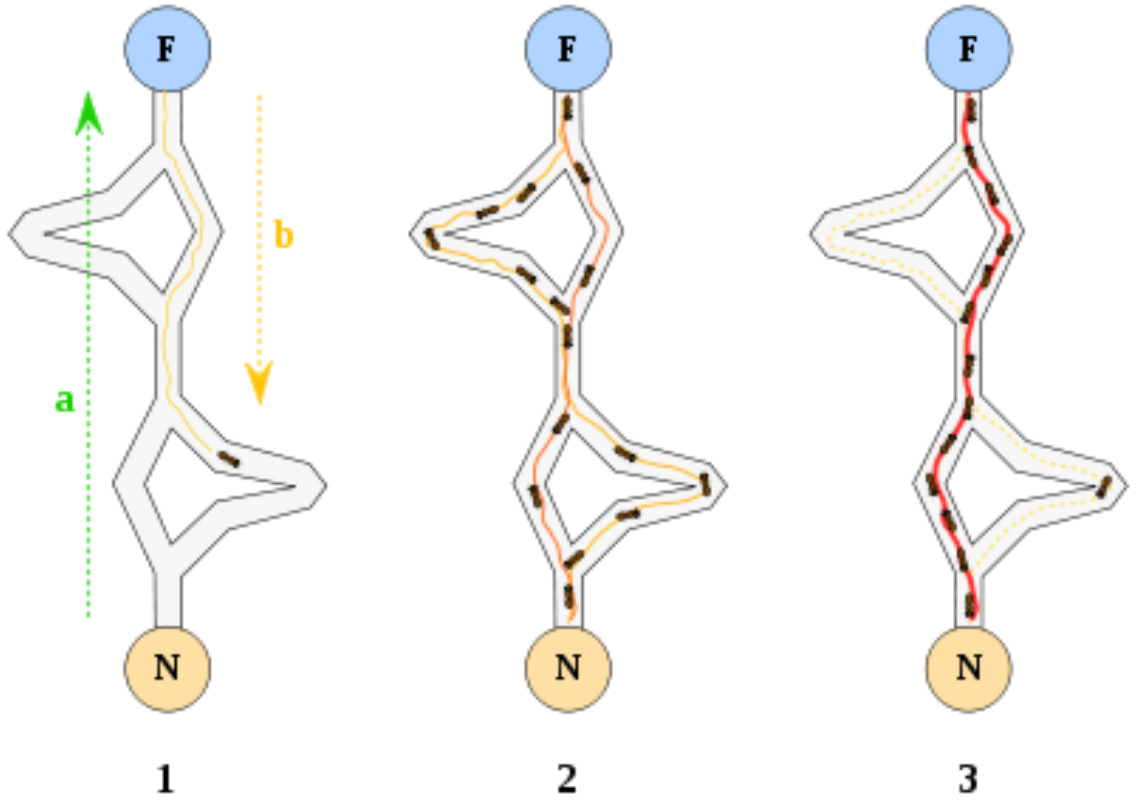
# Ant Colony Optimization (ACO): Basics

- The ants are BLIND (not all of them).
- The ants are attracted by pheromones in neighbor nodes.
- Once an ant leave the nest start leaving pheromone on the way it walked.
- When they find the food, they come back, placing pheromone again.
- The next time they leave the nest they will take the path with more pheromone.

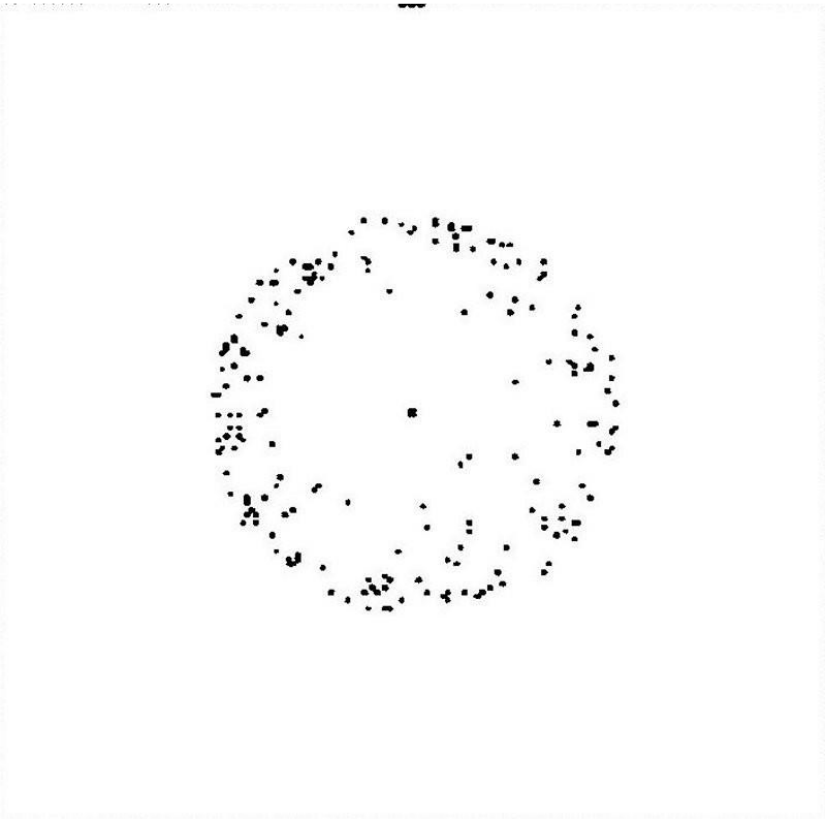
# Ant Colony Optimization (ACO): example



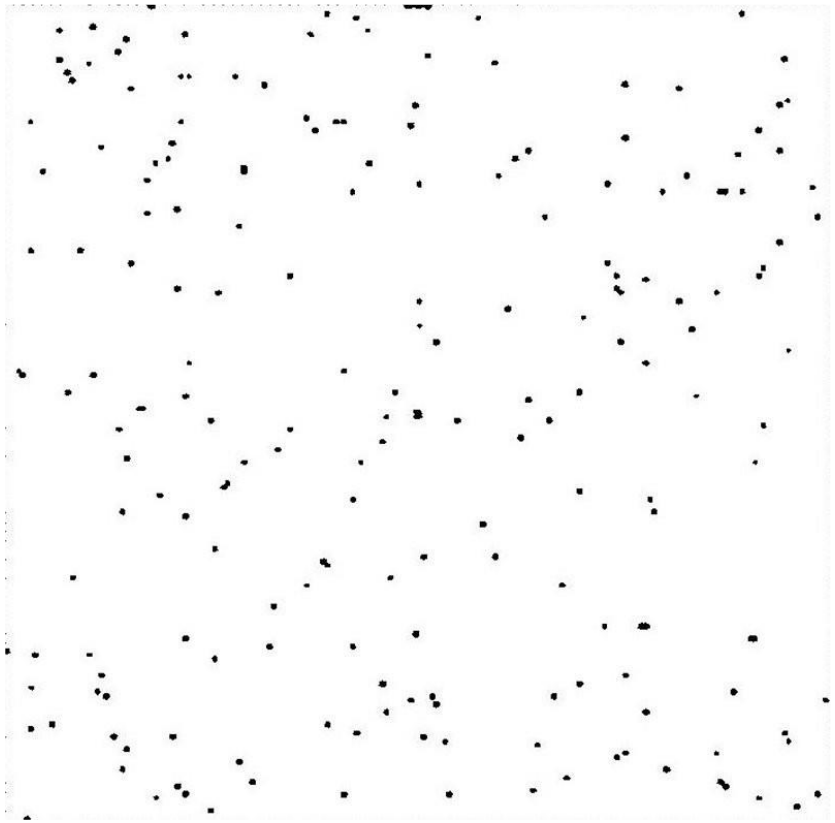
# Ant Colony Optimization (ACO): example



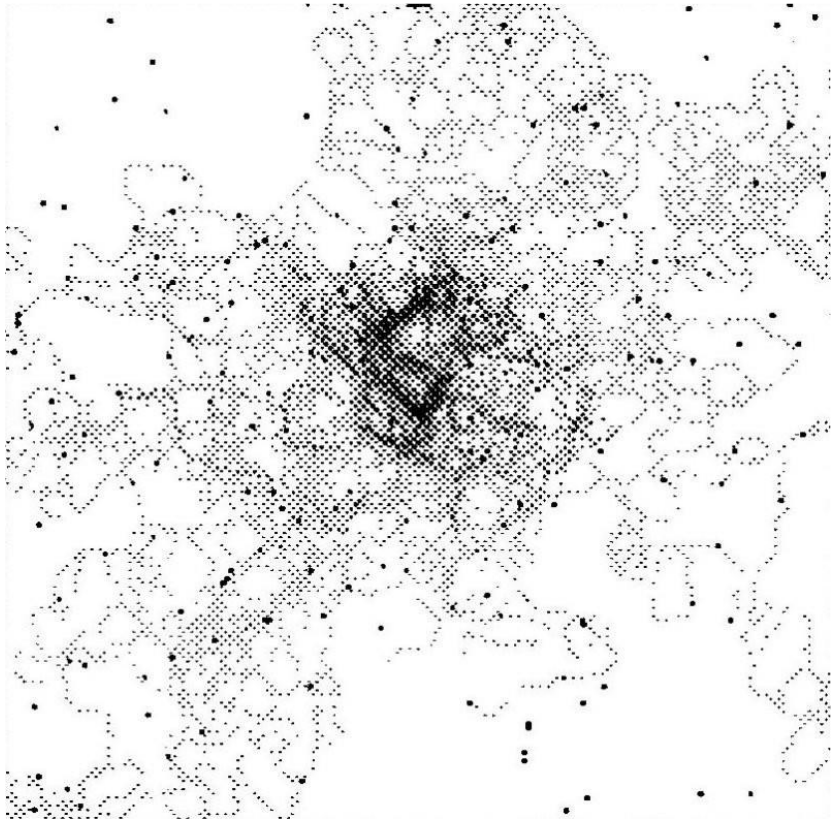
# Ant Colony Opimization (ACO): Natural ants example



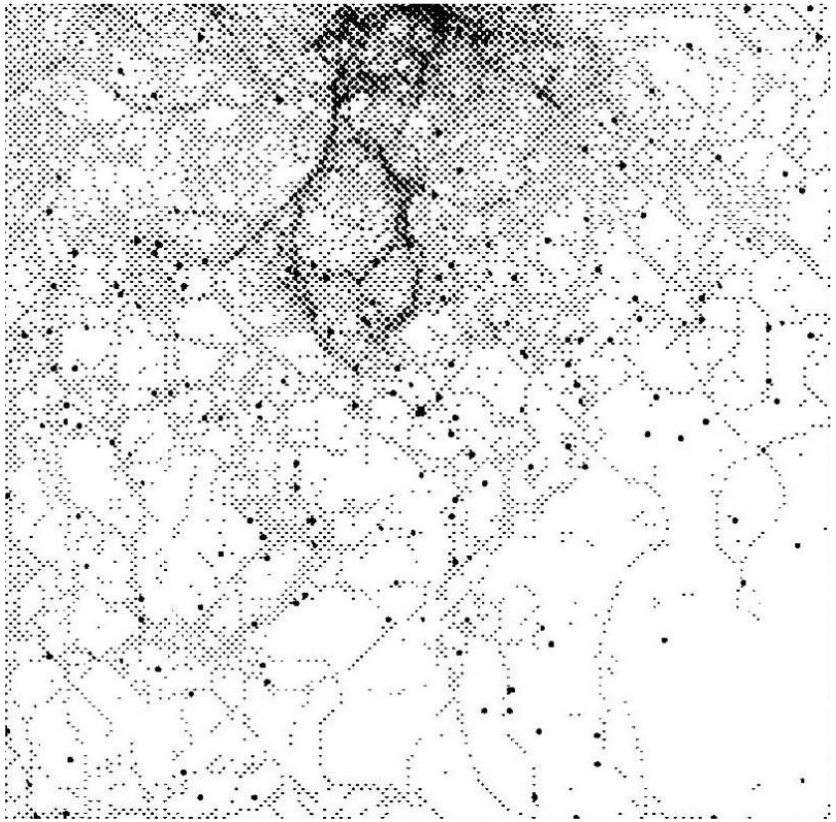
# Ant Colony Opimization (ACO): Natural ants example



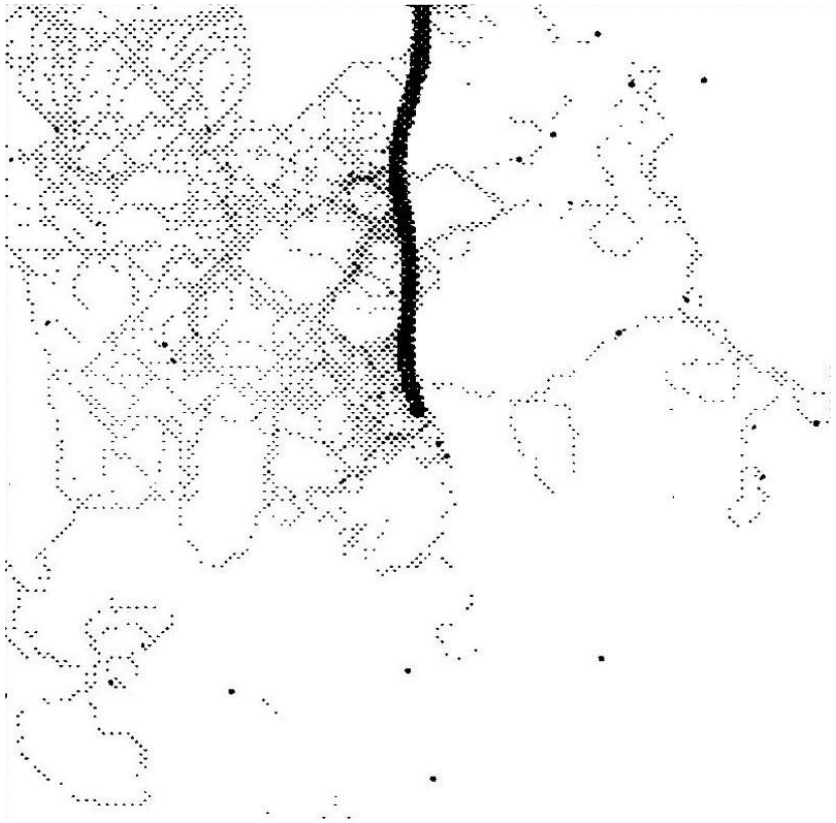
# Ant Colony Opimization (ACO): Natural ants example



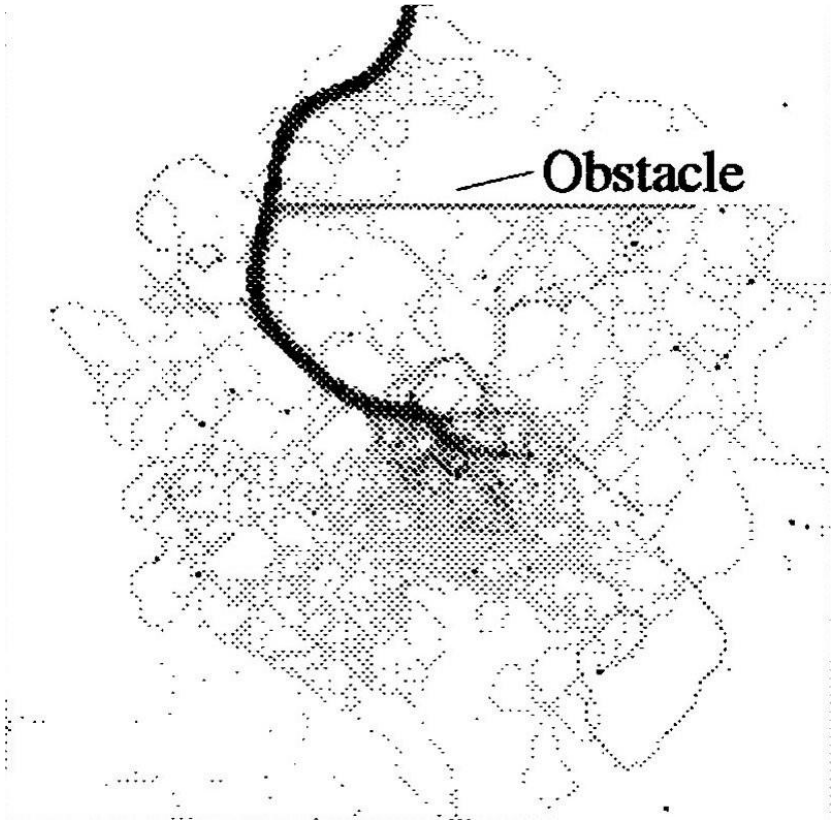
# Ant Colony Opimization (ACO): Natural ants example



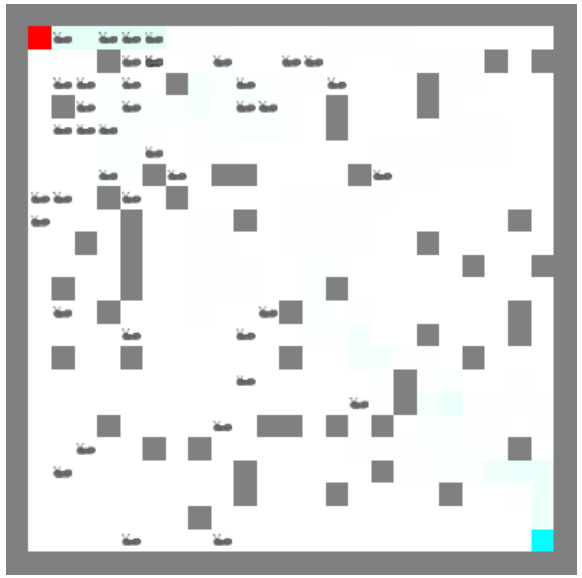
# Ant Colony Opimization (ACO): Natural ants example

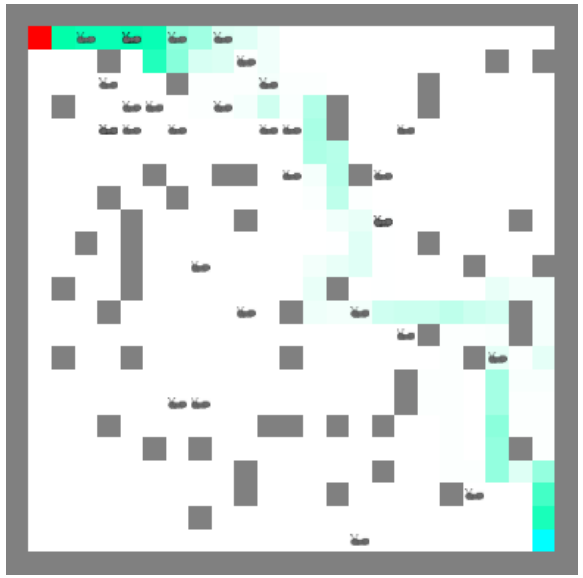


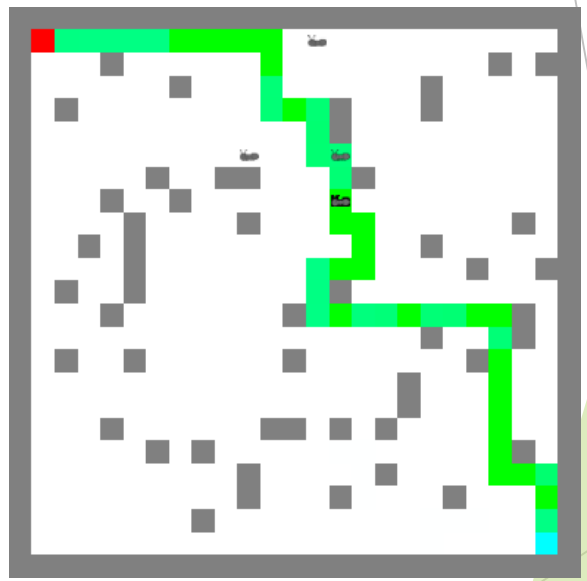
# Ant Colony Opimization (ACO): Natural ants example



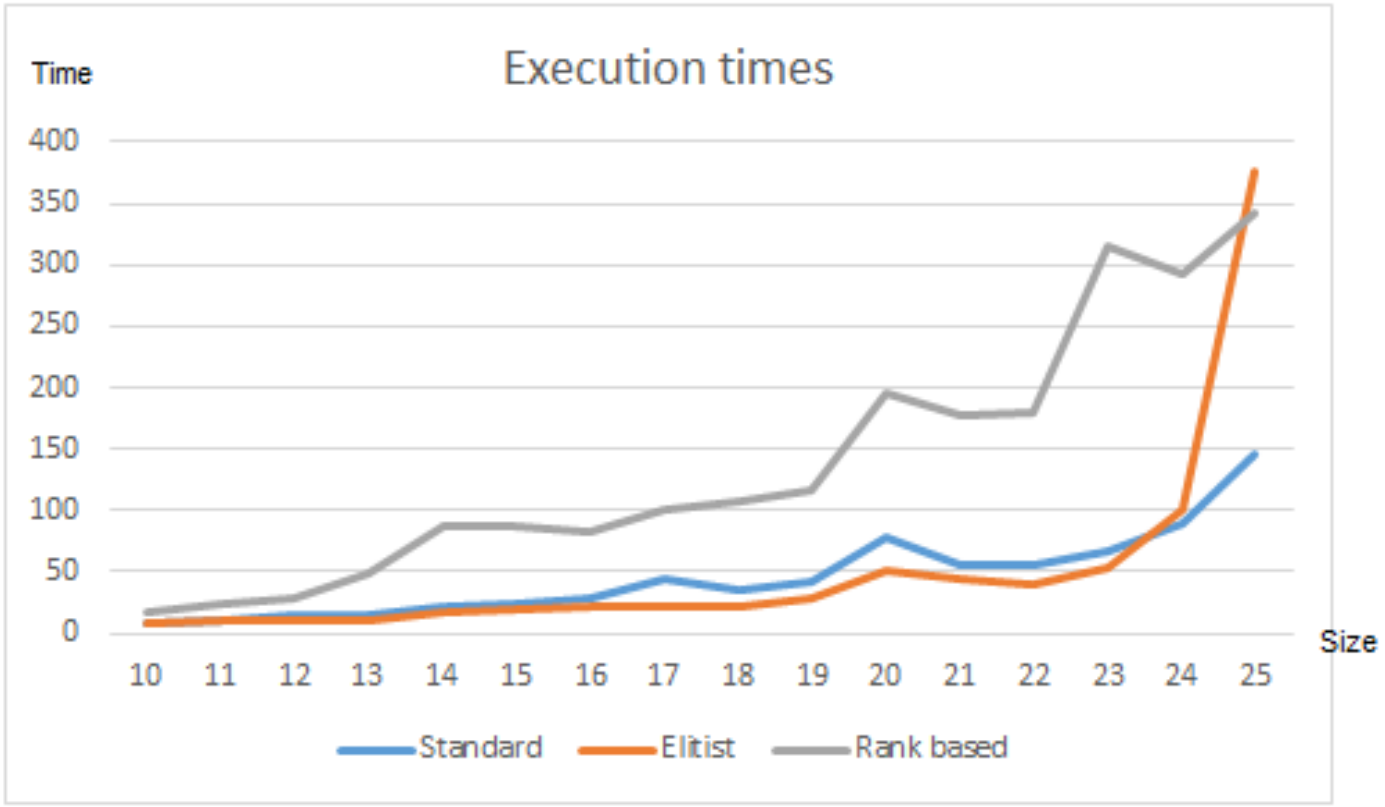
# Ant Colony Opimization (ACO): Project Intelligent system 2015



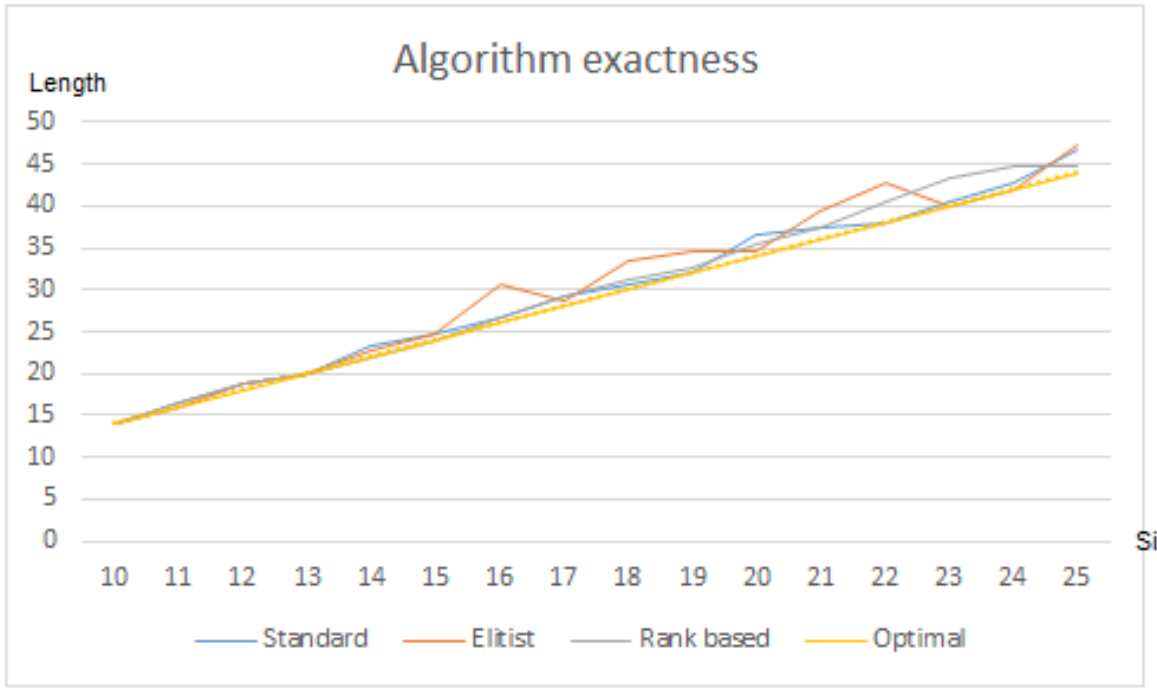




# Ant Colony Opimization (ACO): Project Intelligent system 2015



# Ant Colony Opimization (ACO): Project Intelligent system 2015




ITS!

# Reading Guidance

- It is important to read carefully and understand the content of the slides
- Read the following sections of Chapter of the Machine Learning book of Tom M. Mitchell: 9.1, 9.2, 9.3, 9.5.1

# References


- GA:
  - Tom M. Holland. Machine Learning. 1997
  - Stuart Russell and Peter Norvig. Artificial intelligence: A modern approach. Third version
- DE:
  - Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*(4), 341-359.
- PSO:
  - Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. In *evolutionary computation, 2001. Proceedings of the 2001 Congress on* (Vol. 1, pp. 81-86). IEEE.
- ACO:
  - M. Dorigo, V. Maniezzo, A. Colorni. Ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and cybernetics, Part B, Vol 26, 29-41