Intelligent Systems

UCA Universidad de Cádiz

Unit 3
Population-based Metaheuristics

Dra. Elisa Guerrero Vázquez
elisa.guerrero@uca.es
University of Cadiz - Spain

Contents

1. Population-based Metaheuristics

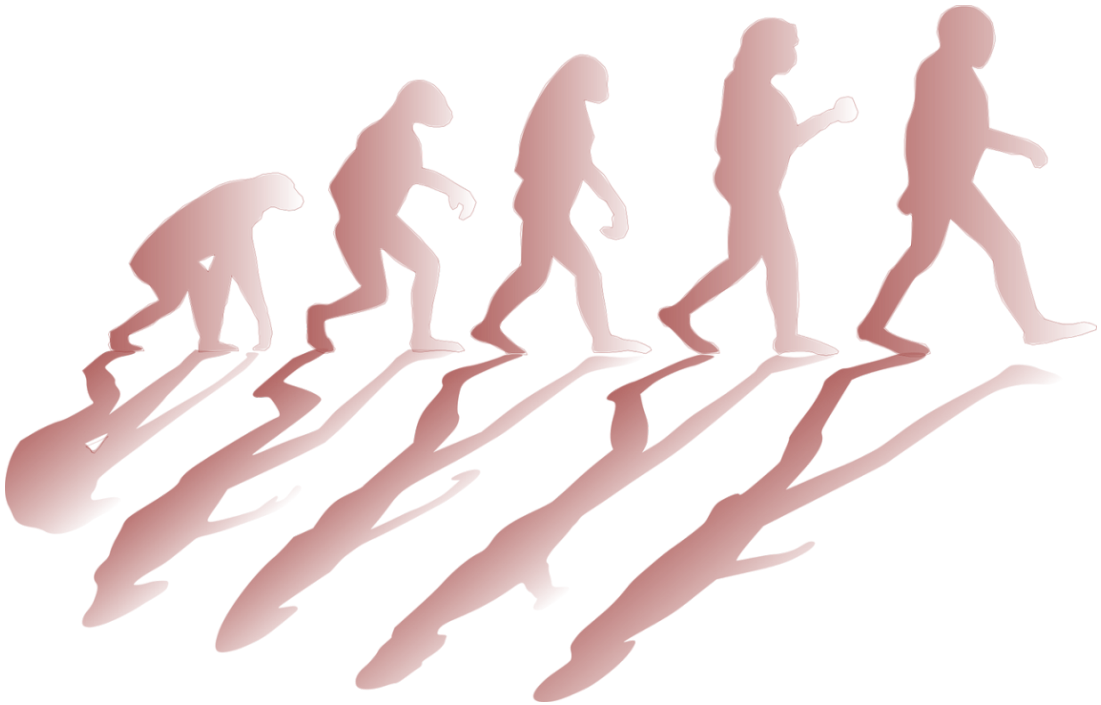
- Metaheuristic: iterative process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions.

  But … There is not guarantee of finding the optimal solution

- Unlike trajectory-based metaheuristics that rely on a single solution as the basis for future exploration, population-based strategies maintain a set of **candidate solutions** in each iteration.

- Most popular approaches:
  - **Genetic Algorithms** (GAs, Holland 70's)
  - **Particle Swarm Optimization** (PSOs, Kennedy, Eberhart and Shi, 90's)
  - Ant Colony optimization (ACO, Dorigo 1992, Hoos and Stützle 1996, Dorigo and Gambardela 1997)
  - Etcetera

2. Genetic Algorithms

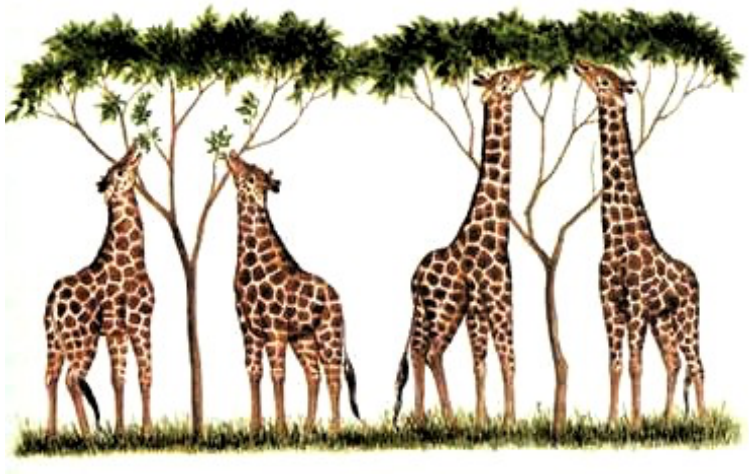


2.1 Introduction

Inspired by Charles Darwin's theory of natural evolution

Individuals in a population compete for limited resources:

- This competition yields to the selection of those individuals better adapted to the environment.
- Genes from the selected individuals propagate throughout the population so that two good parents will sometimes produce offspring that can be better than them.



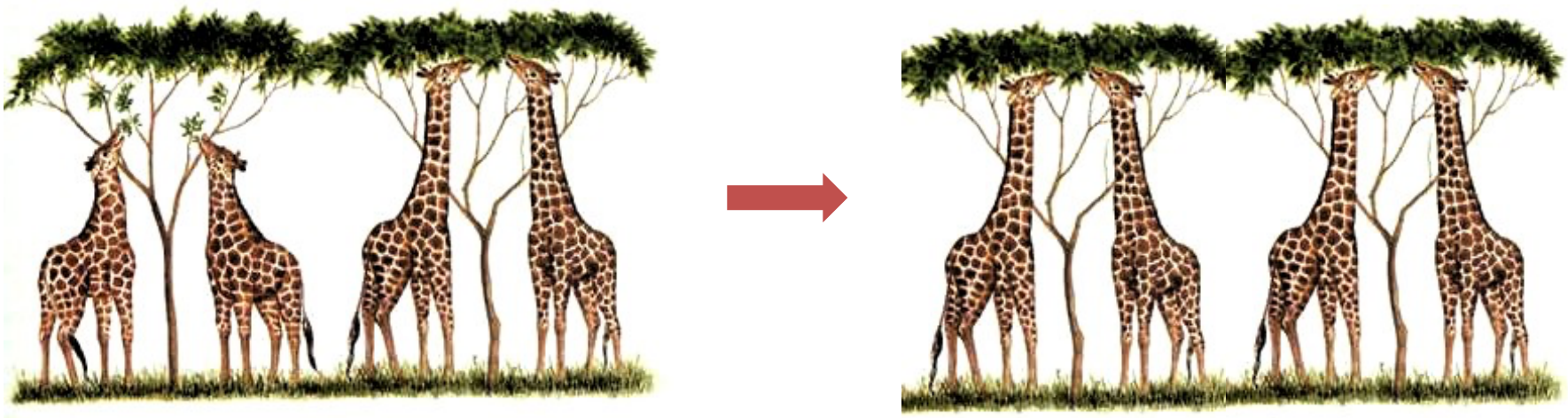
http://naturalesceuja16.blogspot.com/2017/01/la-seleccion-natural-como-funciona-la.html

2.1 Introduction

Inspired by Charles Darwin's theory of natural evolution

Individuals in a population compete for limited resources:

- Thus each successive generation will become more suited to their environment.
- Throughout time, the process of natural selection will allow a better adaptation of the entire population to the environment.



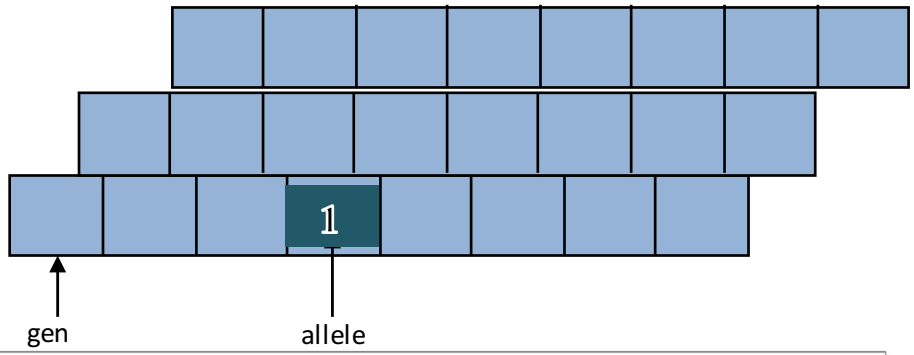
http://naturalesceuja16.blogspot.com/2017/01/la-seleccion-natural-como-funciona-la.html

2.2 Core glossary

Population: set of individuals

Generation: population in a certain iteration

Individual: candidate solution (chromosome, string)



Gene: each chromosome is divided into genes: parameter (variable) that describes the individual

Allele: It is the value a gene takes

2.3 Chromosome Representation

- Binary
  - E.g. SAT Problem

x_1	x_2	x_3	x_4	x_5
0	1	1	1	0

- Integer
  - E.g. N-Queens problem: each gen represents a queen and each allele the row

Q1	Q2	Q3	Q4
4	3	1	2

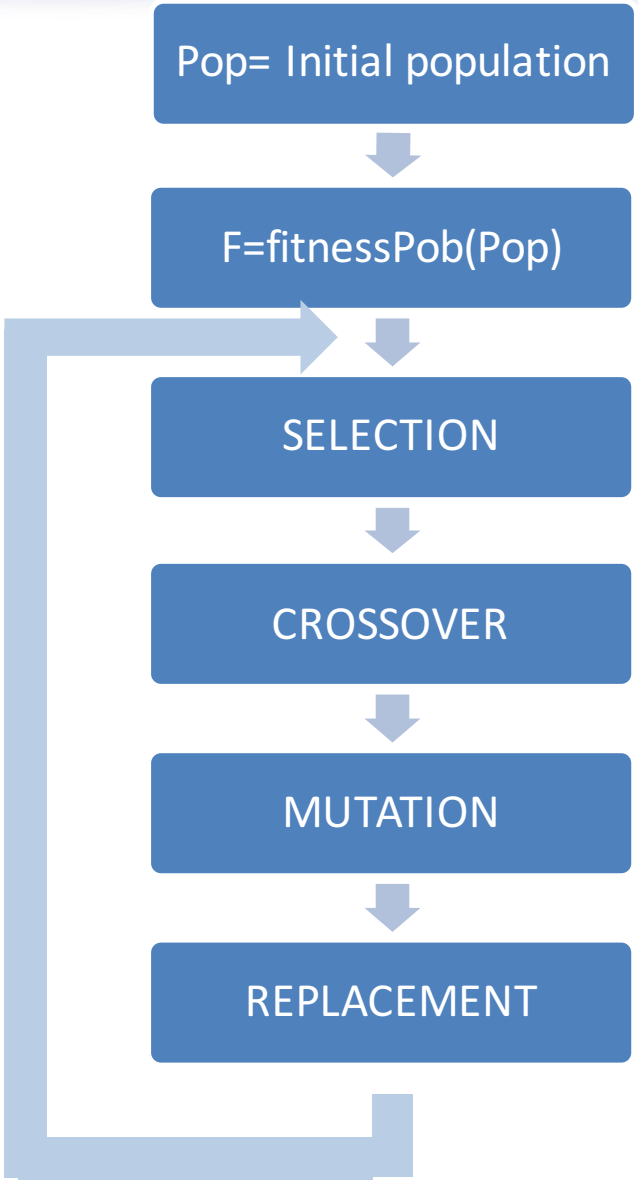
- Floating point
  - E.g. Function optimization

x_1	x_2	x_3	x_4	x_5
0.34	1.25	11.8	5.67	0.98

2.4 Overall process

- The overall process consists of iterating a certain number of times, maintaining a population of potential solutions, in each iteration:
  - The fittest individuals are selected in order to produce a new generation
  - Some of the offspring suffer from alterations due to mutation, crossover, etc.
  - As the execution progresses, the overall fitness of the population improves until reaching a stop criterion

2.5 Operators



- Random generation of a population of candidate solutions
- **EVALUATE** each individual
- **SELECTION:** choose the best candidates and form pairs of parents
- **CROSSOVER:** combine the genes of each pair to obtain two new offsprings from two parents
- **MUTATION:** alter some allele
- **REPLACEMENT:** generate a new population

2.4 Overall process

- Initial random population

	Individuals	Fitness
C1	2 1 4 3	2
C2	1 3 4 2	5
C3	1 2 4 3	4
C4	3 2 4 1	5
C5	4 3 2 1	0
C6	2 3 1 4	5

in maximization problems, the higher values the better.

2.4 Overall process

Selection

	Individuals	Fitness
C1	2 1 4 3	2
C2	1 3 4 2	5
C3	1 2 4 3	4
C4	3 2 4 1	5
C5	4 3 2 1	0
C6	2 3 1 4	5

No selected to be combined

No selected to be combined

2.4 Overall process

Crossover

Parents selected to form couples

C2	1	3	4	2
C6	2	3	1	4

C3	1	2	4	3
C4	3	2	4	1

C2	1	3	4	2
C3	1	2	4	3

Generated Offsprings

O1	1	3	4	2
O2	4	3	1	2

O3	3	2	4	1
O4	1	2	4	3

O5	2	3	4	1
O6	3	2	4	1

2.4 Overall process

Crossover

C2	1	3	4	2
C6	2	3	1	4

C3	1	2	4	3
C4	3	2	4	1

C2	1	3	4	2
C3	1	2	4	3

Generated Offsprings

O1	1	3	4	2
O2	4	3	1	2

O3	3	2	4	1
O4	1	2	4	3

O5	2	3	4	1
O6	3	2	4	1

Mutation

O2	1	3	4	2

O6	3	4	2	1

2.4 Overall process

Replacement

Current and New Individuals

	Individuals	Fitness
C1	2 1 4 3	2
C2	1 3 4 2	5
C3	1 2 4 3	4
C4	3 2 4 1	5
C5	4 3 2 1	0
C6	2 3 1 4	5
O1	1 3 4 2	5
O2	1 3 4 2	5
O3	3 2 4 1	5
O4	1 2 4 3	4
O5	2 3 4 1	2
O6	3 4 2 1	4

New Generation

Cnew1	1 3 4 2
Cnew2	2 3 1 4

Cnew3	1 3 4 2
Cnew4	1 2 4 3

Cnew5	3 2 4 1
Cnew6	2 3 4 1

2.5.1 Selection

To choose the parents that will be combined:

- The key idea is to give preference to better individuals, allowing them to pass on their genes to the next generation.
- Individuals with higher fitness should have more chance to be selected
- Some individuals will be selected more than once, while others will die without leaving any descent.

2.5.1 Selection

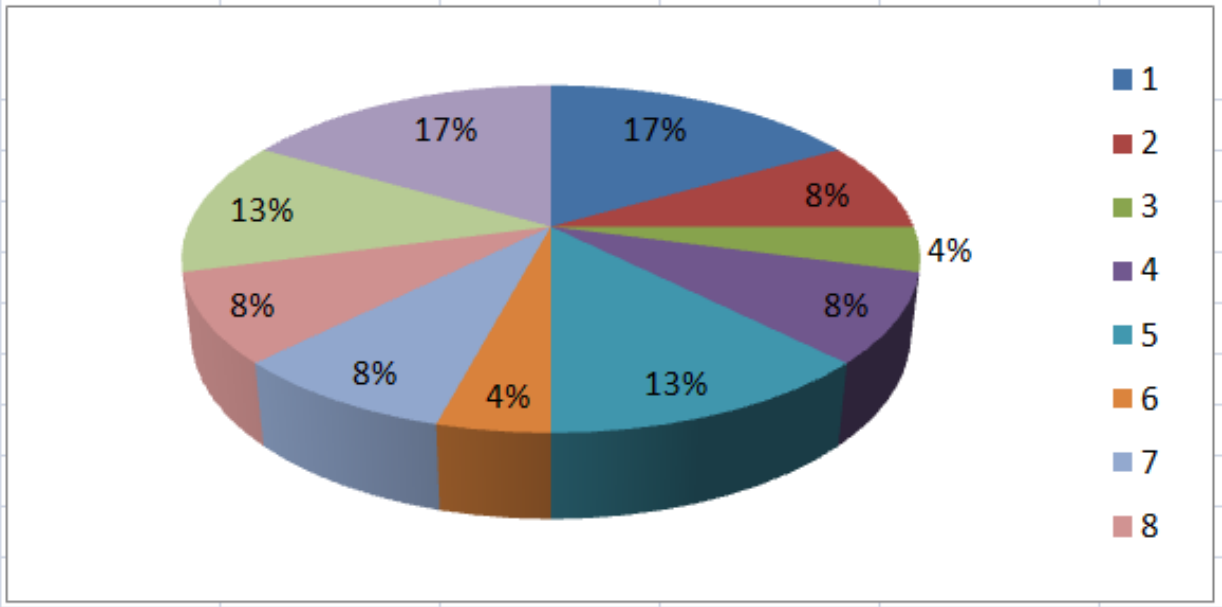
To know which individuals will be recombined:

- **Proportional Selection (Roulette-wheel)**: The individual's probability of being chosen depends on its fitness value with respect to the average fitness of the population.
- **Ranking Selection**: individuals with better fitness values are selected. The better fitness the more offspring.
- **Tournament**, Pick *k* members at random, then select the best of these subset. Repeat the process *n* times, where *n* is the population size.

Roulette Wheel Selection

- Assign to each individual a part of the roulette wheel, based on the evaluation function
- Spin the wheel n times to select n individuals

Función de Evaluación	Prob	Acumulada
4	0,17	0,17
2	0,08	0,25
1	0,04	0,29
2	0,08	0,38
3	0,13	0,50
1	0,04	0,54
2	0,08	0,63
2	0,08	0,71
3	0,13	0,83
4	0,17	1,00



Roulette Wheel Selection

- **Roulette Construction**: generate a vector R containing the cumulative probability distribution for each individual. In this way, the individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness.
- **Probabilities generation:** A set P of N random numbers is generated, representing the probability of choosing an individual.
- **Selection:** The individual whose segment includes the random number is selected. The process is repeated for each element of P

Tournament

1. Select k individuals (randomly) from the population and perform a tournament amongst them
2. Select the best individual from the k individuals
3. Repeat process 1 and 2 until you have the desired amount of population

	Fitness
C1	2
C2	5
C3	4
C4	5
C5	0
C6	5

K=3

C1	2
C4	5
C5	0

The best from this pool

C2	5
C3	4
C6	5

The best from this pool

C1	2
C3	4
C5	0

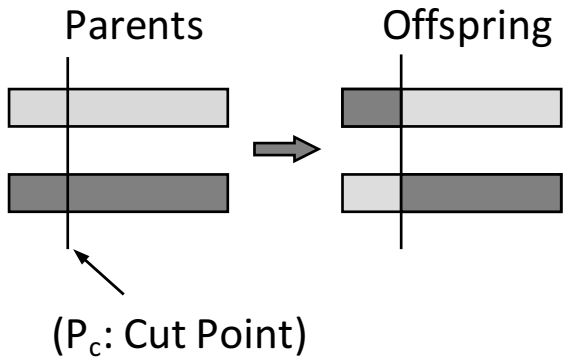
The best from this pool

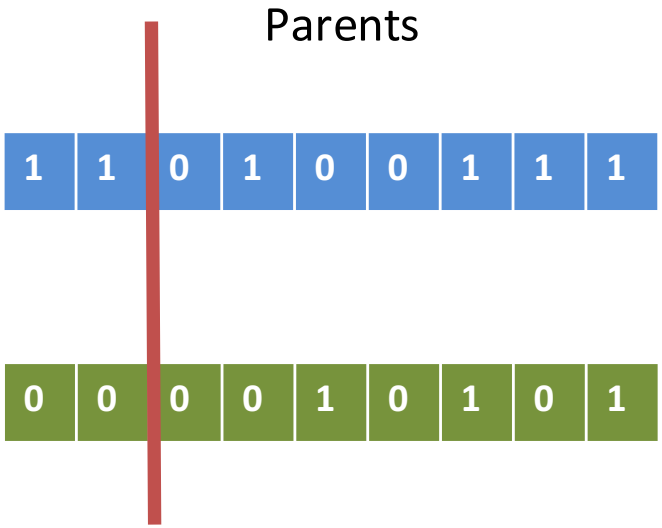
2.5.2 Crossover

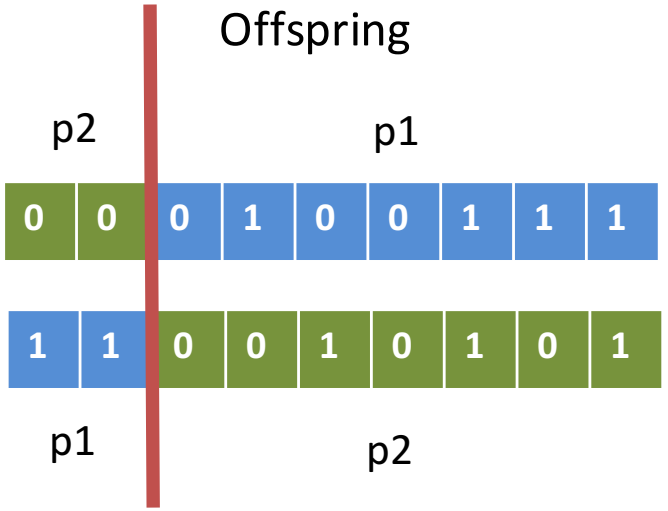
- Two individuals are chosen from the population using the selection operator.
- The two new offspring created from this mating are put into the next generation of the population.
- By recombining portions of good individuals, this process is likely to create even better individuals.
- Depending on the representation there exist different methods:
  - k-point crossover
  - Uniform crossover
  - PMX
  - OX
  - Etc.

1-point crossover

- **Choose a random point on the two parents, P_c**
  - Split parents at this point
  - Create children by exchanging tails

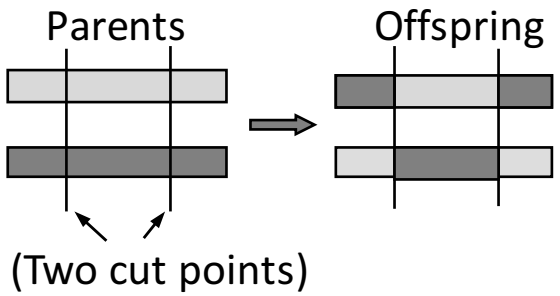


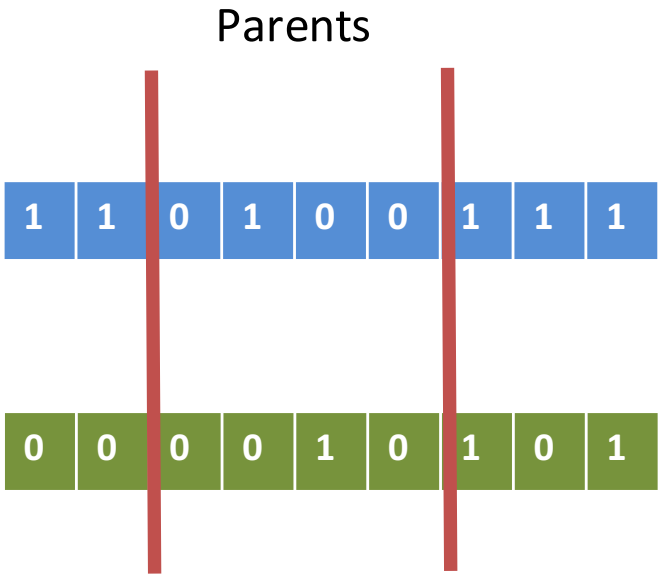


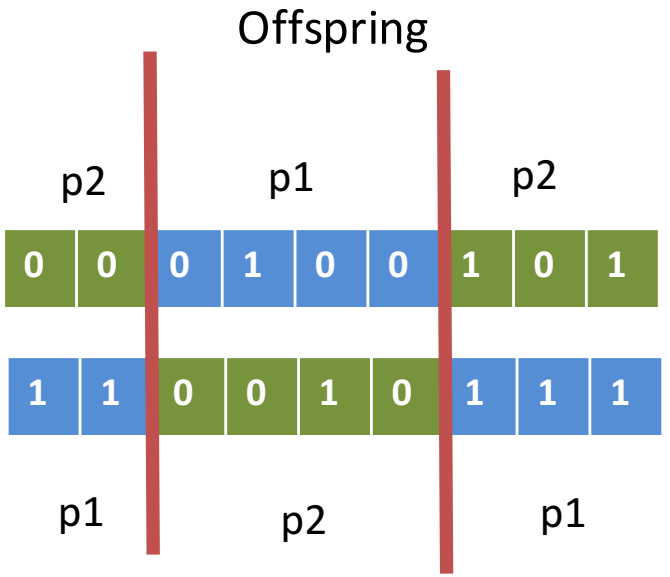


2-points crossover

- **Crossover based on two points**

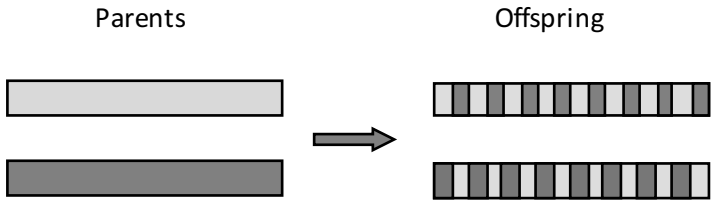






Uniform crossover

- **Uniform crossover**: A random binary mask indicates what gen take from each parent



Crossover mask 1 1 2 1 2 2 1 2

Parent 1 **1 1 1 0 0 0 0 1** Parent 2 **1 0 1 1 1 0 0 0**

Offspring 1 **1 1 1 0 1 0 0 0**

Partially Matched Crossover (PMX)

- Tries to preserve order and position.

1. Two points are selected at random (or determined before execution)

p_1 = (1 2 3 | 4 5 6 7 | 8 9) p_2 = (4 5 2 | 1 8 7 6 | 9 3)

2. The central part of one parent is mapped to the central area of the other parent, keeping the interchanges: 1/4, 8/5, 7/6, y 6/7

o_1 = (x x x | 1 8 7 6 | x x) o_2 = (x x x | 4 5 6 7 | x x)

3. Then, the values that are not in conflict (already inserted) are added to each offspring:

For example value 1 in p1 already exists in s1, then we look for the next value

o1 = (x 2 3 | 1 8 7 6 | x 9) o2 = (x x 2 | 4 5 6 7 | 9 3)

4. Finally, the values in conflict must be replaced by the interchanges: 1/4, 8/5, 7/6, y 6/7. *For example value 1 in p1 already exists in s1, the interchange was 1/4 , thus, value 4 is added instead*

o1 = (4 2 3 | 1 8 7 6 | 5 9) o2 = (1 8 2 | 4 5 6 7 | 9 3)

PMX crossover: some considerations

For example: [1 | 2 3 | 4]

[2 | 3 1 | 4]

First offspring would be [x|3 1| 4] interchanges: 3/1 2/3

Trying to assign to gen x the value 1 from parent 1, yields to a conflict:
[1|3 1| 4]

and the interchange 3/1, yields to another conflict: [3|3 1| 4] ,

thus next interchange must be considered 2/3 [**2**|3 1| 4]

Order Crossover

- From a substring of p1, and preserving the relative order of the p2:

1. Two points are selected at random (or determined before execution)

p1 = (1 2 3 | 4 5 6 7 | 8 9) p2 = (4 5 2 | 1 8 7 6 | 9 3)

2. The central part is copied into the offspring:

o1 = (x x x | 4 5 6 7 | x x) o2 = (x x x | 1 8 7 6 | x x)

3. Starting from the second point, copy the values of the other parent in the same order, omitting those repeated values.

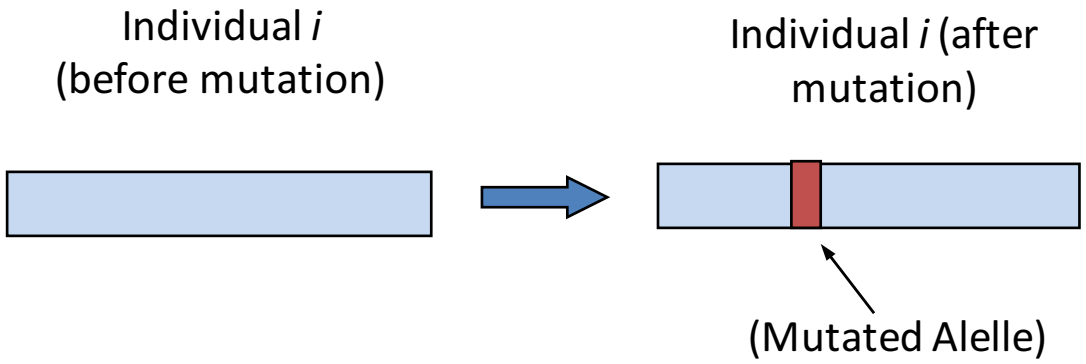
o1 = (x x x | 4 5 6 7 | 9 3) o2 = (x x x | 1 8 7 6 | 9 2)

4. When the end of the string is reached, start from the first position on the left:

o1 = (2 1 8 | 4 5 6 7 | 9 3) o2 = (3 4 5 | 1 8 7 6 | 9 2)

2.5.3 Mutation

- Alter each gene independently with a probability P_{mut}
  - P_{mut} is called the mutation rate
  - Typically between **1/pop_size** and **1/chromosome_length**
- It is a way to promote diversity and to avoid local optima



Mutation operators

- **Bit Flip:** We select one or more random bits and flip them (from 0 to 1 or viceversa)
- **Inversion:** Invert the order of a sub-chain

v = (1 9 8 | **7 6 5 4** | 3 2) → v' = (1 9 8 | **4 5 6 7** | 3 2)

- **Swap**: interchange the values of two genes selected randomly
- **Insertion:**
  1. Pick two allele values at random
  2. Place the first to follow the second, shifting the rest along to accommodate

v = (1 9 **8** 7 6 5 4 3 2) → v' = (1 9 7 6 5 4 **8** 3 2)

2.6 Replacement

- The replacement strategy defines how individual solutions are selected for survival into every new generation, and plays an important role in achieving balance between exploration and exploitation of the algorithm.
- Populations must have fixed size.
  - **Generational**: replaces all the population with the new offspring.
  - **ELITIST**: better individuals always survive and are present in next generation
  - **Steady-State**: selects two parents and create 1-2 offspring which will replace the 1-2 worst individuals in the current population even if the offspring are worse. SSGAs are overlapping systems, because parents and offspring compete for survival.

2.7 Stopping criteria

- Reach a maximum number of generations,
- Reach a certain fitness value that it is considered good enough
- The population fitness has reached a plateau
- Reach a certain amount of time

3. Particle Swarm Optimization

3. PARTICLE SWARM OPTIMIZATION (PSO)

Inspiration in the social interaction between species

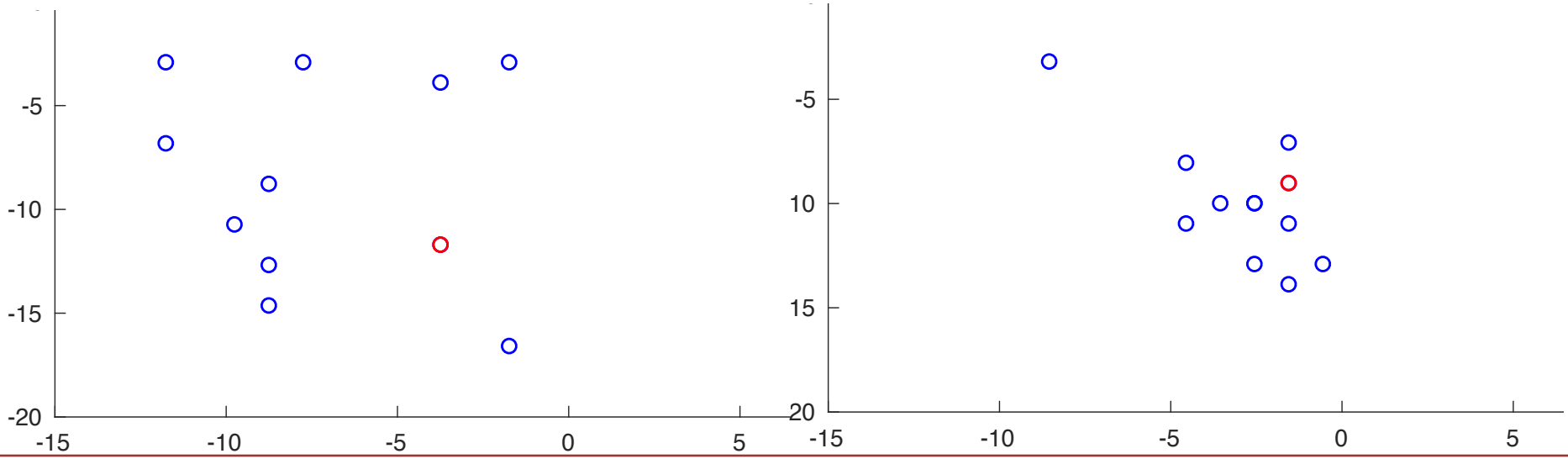
PSO is an optimisation technique based on the movement and intelligence of certain animal populations, such as schools of fish, flocks of birds, etc.

- PSO uses a certain number of agents (particles) that simulate the movement of a population in search of the best solution
- Each particle is treated as a point in N-dimensional space that adjusts its motion according to the experience of its own movement, as well as the experience of other particles

3.1 Basis

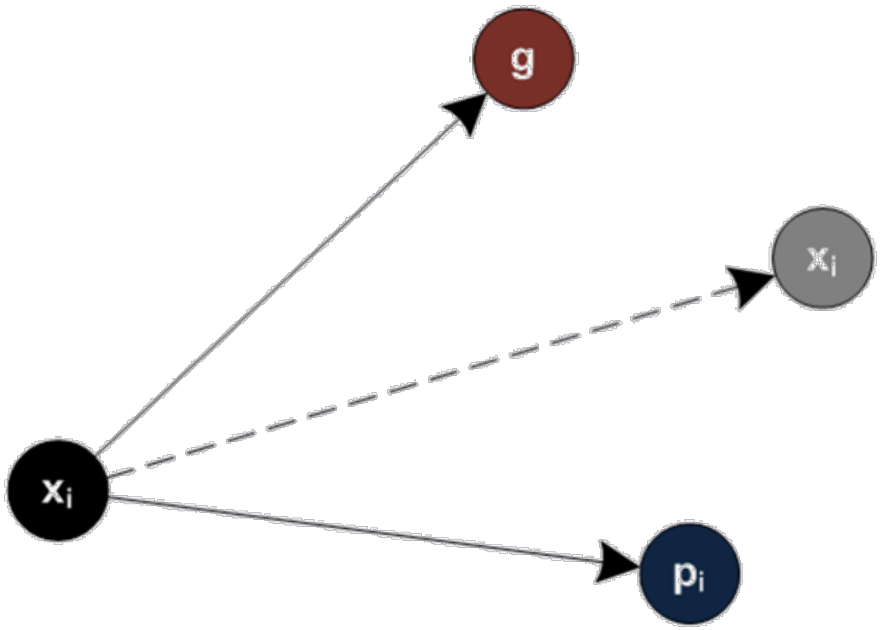
- **Personal best, *pbest*:** Each particle stores its coordinates in the state space. They are associated with the best solutions (fitness) achieved so far by that particle.
- **Global optimal, *gbest*:** The best value obtained by a particle belonging to the whole population.

The basic process consists of guiding each particle towards the ***pbest*** and ***gbest*** locations, by changing the speed of the movement.



3.1 Basis

- A particle moves towards the best position in the search space, remembering:
  - its personal best location reached so far
  - the global best location of a particle in the whole population



3.2 PSO Algorithm

1. Generate randomly a population of different particles with different velocities
2. In each iteration *k:*
   1. Evaluate fitness of each particle, *i*
   2. Determine each pbest and the global gbest
   3. Update velocity and location of each particle, *i*, according to:
      - Current position S_i^k
      - Current velocity V_i^k
      - Distance between the current position and pbest $(S_{pbest_i} - S_i^k)$
      - Distance between the current position and gbest $(S_{gbest} - S_i^k)$

3.2 PSO Algorithm

$$V_i^{k+1} = wV_i^k + c_p\, r_p(S_{pbest_i} - S_i^k) + c_g\, r_g(S_{gbest} - S_i^k)$$

$$S_i^{k+1} = S_i^k + V_i^{k+1}$$

- Parameters:
  - S_i^k Position of the particle i in the iteration k
  - V_i^k Velocity of the particle i in the iteration k
  - S_{gbest} Best global position
  - $S_{pbest\ i}$ Best personal position of the particle i
- Coefficients:
  - Inertial constant ***w***, to establish the influence of the previous solution
  - Accelerating constants **(c_p, c_g)** to establish the influence of the best particular and global
  - Random numbers **(r_p, r_g)**: to avoid local optima

3.3 Updates

- In each iteration, the particle's behaviour is a trade-off among three possibilities:
  - Follow its own exploration pattern
  - Go towards the best personal position
  - Go towards the best global position, following the "leader"

$$V_i^{k+1} = wV_i^k + c_p\, r_p(S_{pbest_i} - S_i^k) + c_g\, r_g(S_{gbest} - S_i^k)$$
$$S_i^{k+1} = S_i^k + V_i^{k+1}$$

3.4 Summarizing

- Each agent or particle:
  - has a position (candidate solution)
  - has a velocity (to change the position more or less quickly)
  - remembers its best personal position
  - remembers the best global position
  - knows how to calculate next position and evaluation function

References

- Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Algorithms* (Springer, 1995). 3ª Edición
- Michalewicz, Z. y Fogel D. *How to solve it. Modern Heuristics* (Springer, 2004)
- El-Ghazali Talbi. *Metaheuristics. From design to implementation* (Wiley 2009)
- Swarm Intelligence: http://www.swarmintelligence.org/
- Particle Swarm Optimization, James Kennedy and Russell Eberhart, 1995
- Russell, S. y Norvig, P. *Inteligencia Artificial (un enfoque moderno) (Pearson Educación, 2004). Segunda edición.*
- García Serrano, A. *Inteligencia artificial. Fundamentos, práctica y aplicaciones* (RC Libros, 2012)
- Tim Jones, J. *Artificial Intelligence. A systems approach.* Computer Sciences Series. (Jones & Bartlett publishers, 2008)