Lesson 2:

How to "build" a Genetic Algorithm

António Gaspar-Cunha

Institute for Polymers and Composites/I3N, Dept. of Polymer Engineering,
University of Minho, Guimarães, Portugal

http://www.dep.uminho.pt/agc/

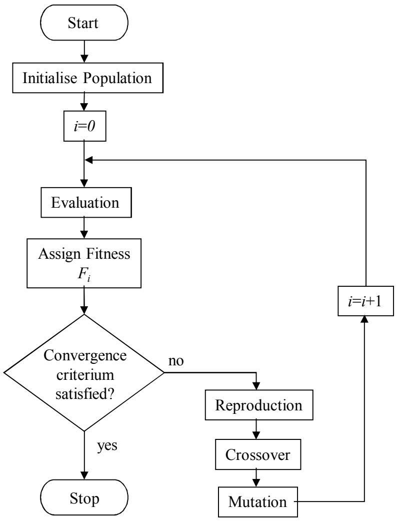
Universidad Carlos III de Madrid
January/February 2012

OUTLINE

- Global Structure of the Algorithm
- Population Initialization
- Evaluation
- Selection
- Mutation
- Recombination (Crossover)
- Inversion
- Final Notes
- Other Algorithms

GLOBAL STRUCTURE OF THE ALGORITHM

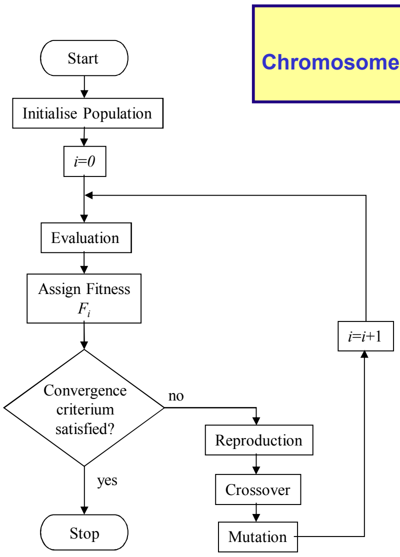
Evolutionary Algorithms

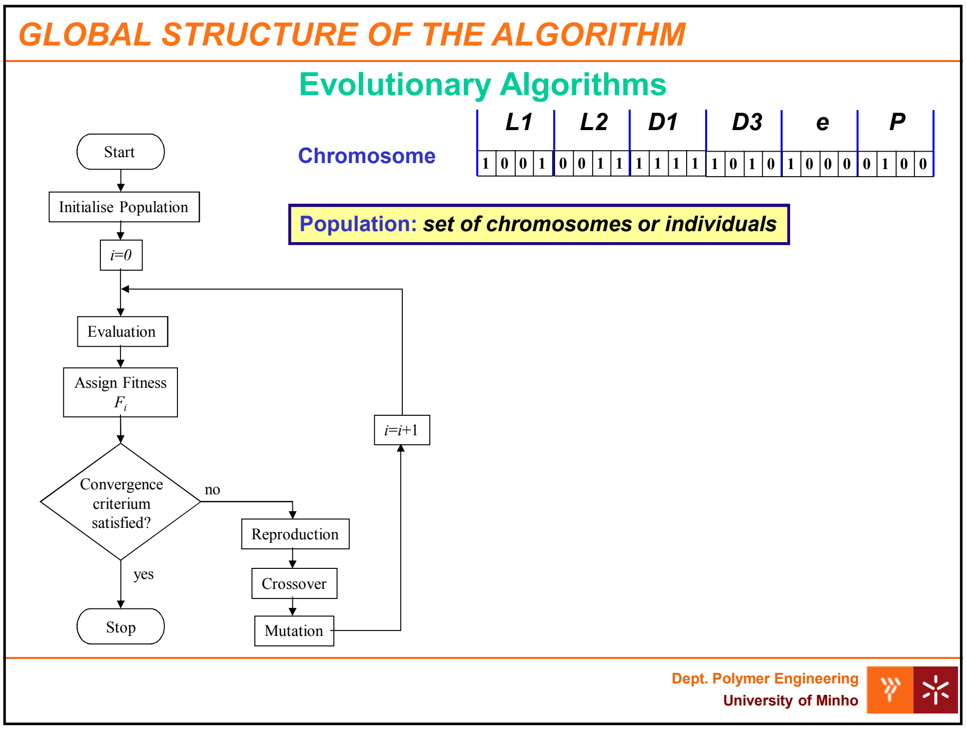


GLOBAL STRUCTURE OF THE ALGORITHM

Evolutionary Algorithms

	L1				**L2**				**D1**				**D3**				**e**				**P**			
Chromosome	1	0	0	1	0	0	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0





GLOBAL STRUCTURE OF THE ALGORITHM

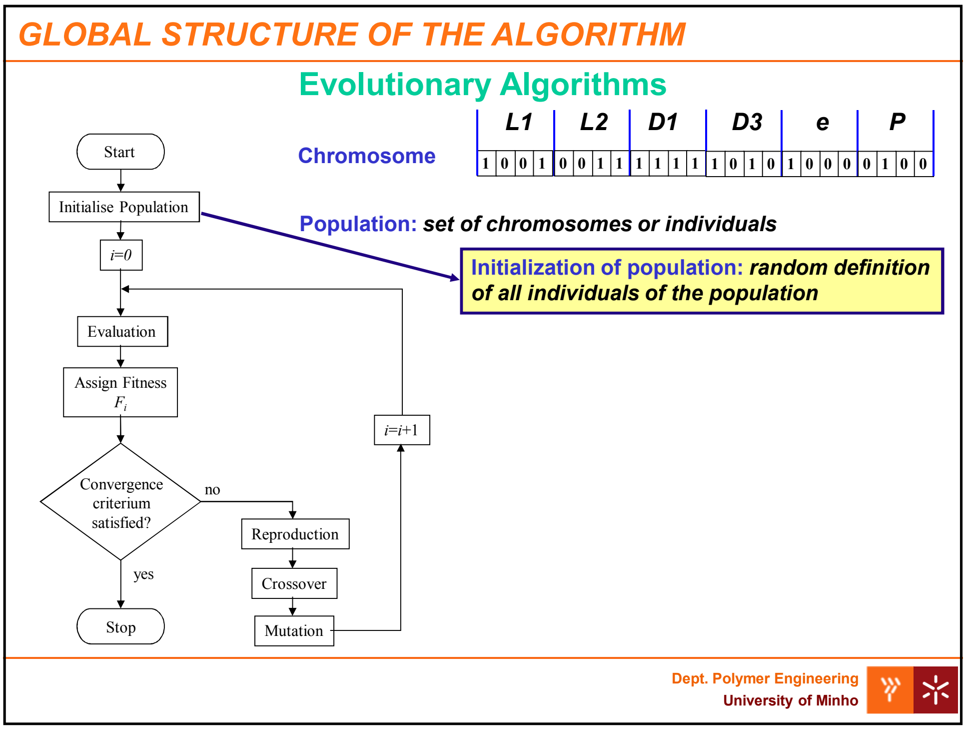
Evolutionary Algorithms

Chromosome

L1	L2	D1	D3	e	P
1 0 0 1	0 0 1 1	1 1 1 1	1 0 1 0	1 0 0 0	0 1 0 0

Population: *set of chromosomes or individuals*

Dept. Polymer Engineering
University of Minho



GLOBAL STRUCTURE OF THE ALGORITHM

Evolutionary Algorithms

Chromosome

L1	L2	D1	D3	e	P
1 0 0 1	0 0 1 1	1 1 1 1	1 0 1 0	1 0 0 0	0 1 0 0

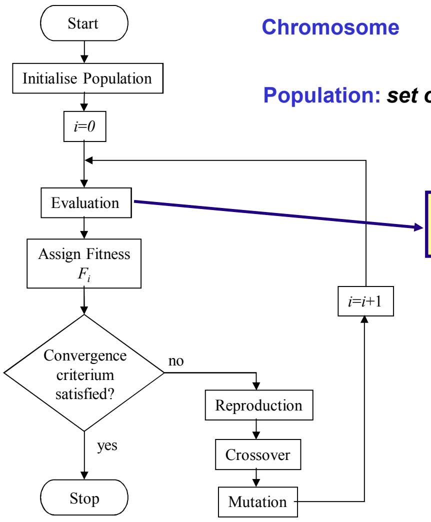
Population: *set of chromosomes or individuals*

Initialization of population: *random definition of all individuals of the population*

Dept. Polymer Engineering
University of Minho

GLOBAL STRUCTURE OF THE ALGORITHM

Evolutionary Algorithms



Chromosome

L1				L2				D1				D3				e				P			
1	0	0	1	0	0	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0

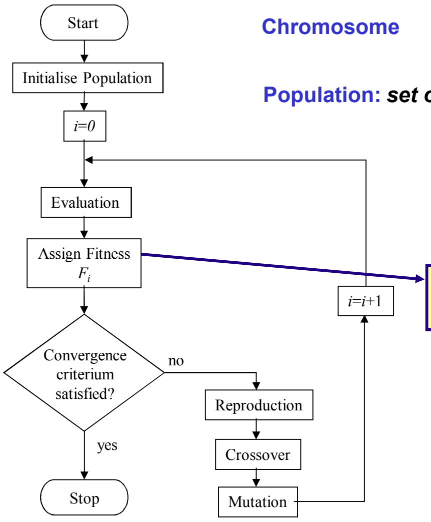
Population: *set of chromosomes or individuals*

Initialization of population: *random definition of all individuals of the population*

Evaluation: *calculation of the values of the criteria using the modeling routine*

GLOBAL STRUCTURE OF THE ALGORITHM

Evolutionary Algorithms



Chromosome

L1				L2				D1				D3				e				P			
1	0	0	1	0	0	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0

Population: *set of chromosomes or individuals*

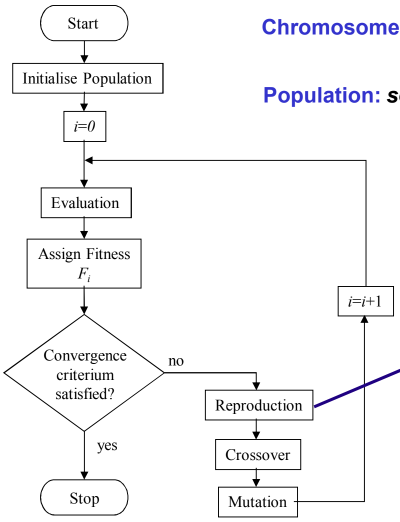
Initialization of population: *random definition of all individuals of the population*

Evaluation: *calculation of the values of the criteria using the modeling routine*

Fitness: *calculation of a single value identifying the performance of individual*

GLOBAL STRUCTURE OF THE ALGORITHM

Evolutionary Algorithms



Chromosome

L1	L2	D1	D3	e	P
1 0 0 1	0 0 1 1	1 1 1 1	1 0 1 0	1 0 0 0	0 1 0 0

Population: *set of chromosomes or individuals*

Initialization of population: *random definition of all individuals of the population*

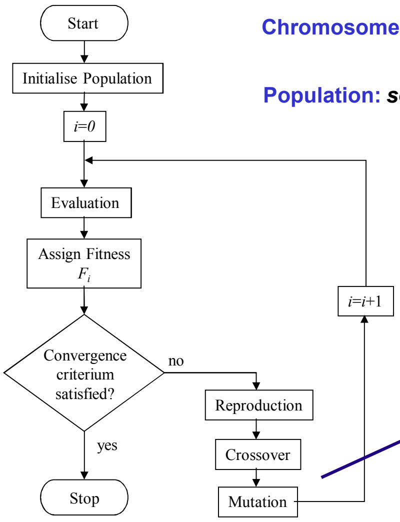
Evaluation: *calculation of the values of the criteria using the modeling routine*

Fitness: *calculation of a single value, identifying the performance of the individual*

Reproduction: *selection of the best individuals for crossover and/or mutation*

GLOBAL STRUCTURE OF THE ALGORITHM

Evolutionary Algorithms



Chromosome

L1	L2	D1	D3	e	P
1 0 0 1	0 0 1 1	1 1 1 1	1 0 1 0	1 0 0 0	0 1 0 0

Population: *set of chromosomes or individuals*

Initialization of population: *random definition of all individuals of the population*

Evaluation: *calculation of the values of the criteria using the modeling routine*

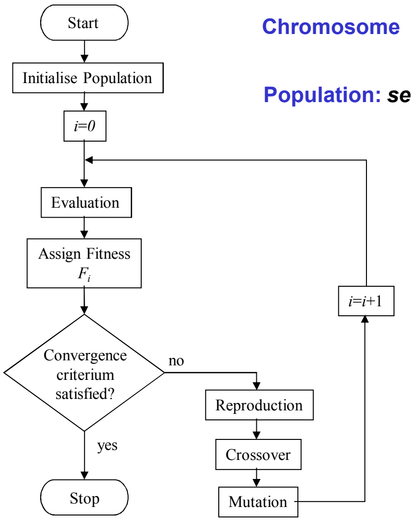
Fitness: *calculation of a single value identifying the performance of individual*

Reproduction: *selection of the best individuals for crossover and/or mutation*

Crossover/Mutation: *methods to obtain new individuals for the next generation (i+1)*

GLOBAL STRUCTURE OF THE ALGORITHM

Evolutionary Algorithms



Chromosome

L1				L2				D1				D3				e				P			
1	0	0	1	0	0	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0

Population: *set of chromosomes or individuals*

Initialization of population: *random definition of all individuals of the population*

Evaluation: *calculation of the values of the criteria using the modeling routine*

Fitness: *calculation of a single value identifying the performance of individual*

Reproduction: *selection of the best individuals for crossover and/or mutation*

Crossover/Mutation: *methods to obtain new individuals for the next generation (i+1)*

GLOBAL STRUCTURE OF THE ALGORITHM

How to Build an Evolutionary Algorithm?

GLOBAL STRUCTURE OF THE ALGORITHM

Stages for designing an EA

- Define/chose a representation to the problem
- Define/chose how to initialise the population
- Define/chose a way of mapping a genotype to a phenotype
- Define/chose a way of evaluating an individual
- Define/chose an appropriate mutation operator(s)
- Define/chose an appropriate recombination operator(s)
- Decide how to select individuals to be parents
- Decide how to manage the population
- Decide when to stop the algorithm

GLOBAL STRUCTURE OF THE ALGORITHM

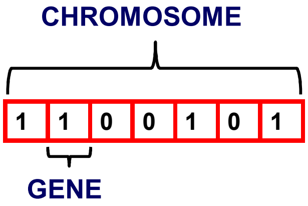
Designing a Representation

- Define a method for representing the individuals as a genotype.
- Since there are several ways to do this, the method selected must take into account the problem to be solved.
- The selection of a given representation depends on how the genotypes will be evaluated and what will be the genetic operators to be used.

GLOBAL STRUCTURE OF THE ALGORITHM

Discrete Representation

- The representation of an individual can be made using discrete values (*binary, integer, or any other system with a discrete set of values*).
- An example of binary representation:



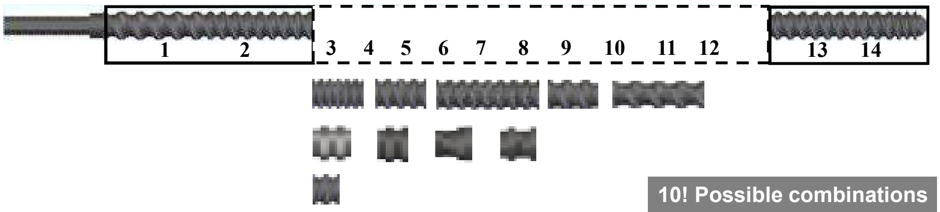
CHROMOSOME

1	1	0	0	1	0	1

GENE

GLOBAL STRUCTURE OF THE ALGORITHM

OBJECTIVE: **To define the best location of available screw elements in order to maximize the process performance**

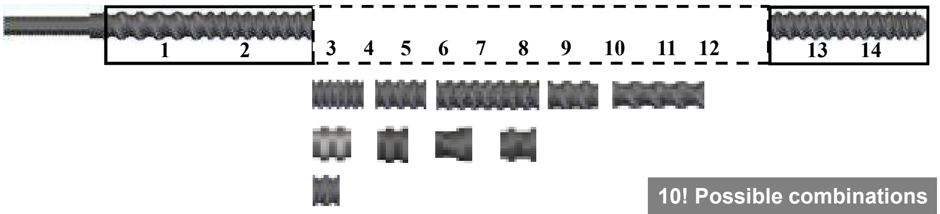


10! Possible combinations

3.6×10^6 configurations

GLOBAL STRUCTURE OF THE ALGORITHM

OBJECTIVE: **To define the best location of available screw elements in order to maximize the process performance**



10! Possible combinations

3.6 x 10^6 configurations

Chromosome: | 1 | 3 | 5 | 6 | 2 | 4 | 7 |

GENES: Location of screw elements

PROBLEM CHARACTERISTICS:

- Discrete decision variables (*position of the screw elements*)

GLOBAL STRUCTURE OF THE ALGORITHM

Discrete Representation

For example, if the phenotype represents a schedule

(e.g. 8 jobs, 2 time steps)

Genotype:

1	0	1	0	0	0	1	1

=

Job	Time Step
1	2
2	1
3	2
4	1
5	1
6	1
7	2
8	2

Phenotype

Chromosome	time step
0	1
1	2

GLOBAL STRUCTURE OF THE ALGORITHM

Real Representation

- If the solution to be obtained is a list of real numbers, naturally the genotype can be a list of these real value numbers
- There are *plenty* of applications

GLOBAL STRUCTURE OF THE ALGORITHM

Real Representation

- If the individuals are represented as a set of n real numbers:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, x_i \in R$$

- The fitness function might map X to a single real number:

$$f : R^n \to R$$

POPULATION INITIALIZATION

Initialization

- Uniformly (randomly) on the search space
  - Binary strings: 0 or 1 with probability 0.5 (each)
  - Real representations, generated uniformly on a pre-defined interval
- Define the population using previous results or specific heuristics to generate it:
  - Could be very useful if the solutions used are a good approximation of the solution
  - Possible loss of genetic diversity, which can imply the possibility of losing some local optima

---

POPULATION INITIALIZATION

Random number generation

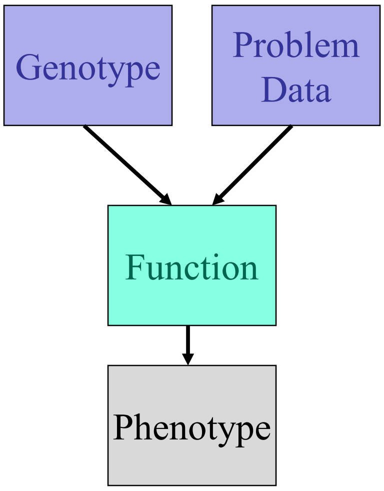
- The use of the normal random number generators existent in the computer systems is not recommended;
- The random number generator must be able to generate the same sequence in successive runs, only depending on the set of the “seed” value;
- Some good random number generators are available (see Numerical Recipes book).

POPULATION INITIALIZATION

From Phenotype to Genotype

Two possibilities:

- Getting the phenotype from the genotype is made directly.
- The genotype is obtained through a function (or algorithm), which may depends on some data.



Genotype

Problem Data

Function

Phenotype

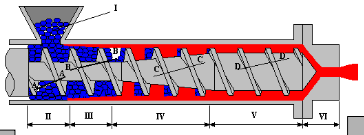
EVALUATION

Evaluating an Individual

- This is the most costly step for real applications, thus it is important that *do not re-evaluate unmodified individuals;*
- Often the individuals are evaluated using a subroutine, a black-box simulator, or other external process;
- There is the possibility of using approximate fitness, however they cannot be used in all generations (we will came back later to this point).

EVALUATION

Polymer extrusion example



***Input data*:**

- Geometry
- Polymer properties
- Operating conditions

Governing equations

***Response*:**

- Output
- Power consumption
- Melt temperature
- Degree of mixing
- ...

EVALUATION

Polymer extrusion example

[redacted]

***Input data*:**

- Geometry
- Polymer properties
- Operating conditions

***Response*:**

- Output
- Power consumption
- Melt temperature
- Degree of mixing
- ...

Specific problems during evaluation

- Constraint handling - what if the phenotype breaks some constraint of the problem:
  - *penalize the fitness*
  - *specific evolutionary methods*
  - *eliminate the solution*
- In the case of multi-objective evolutionary optimization the result is a set of compromise solutions

Selection Strategy

Aim of the selection:

- Give the “better” individuals to have a high chance of being parents. This increase the selection pressure, driving the population forward.

Simultaneously:

- Some attention must be paid to the less good individuals, since they may include some useful genetic material.

Fitness proportionate selection

- The expected number of times f_i is selected for mating is: $f_i / \bar{f}$
- Better (fitter) individuals have:
  - more space
  - more chances to be selected

Fitness proportionate selection

Disadvantages:

- Danger of premature convergence because the worst individuals can be eliminated from the population very fast
- Low selection pressure when fitness values are near each other

SELECTION

Roulette-Wheel selection

This is the simplest selection technique, and consists on:

1. Ordering the population individuals by descending order of their objective function value;
2. Calculating their cumulative sum;
3. Generating a random number between 0 and the total of the above sum;
4. Selecting an individual whose cumulative sum will be closer but greater than the number generated previously;
5. Repeating steps 3 and 4 until the total number of individuals is selected.

SELECTION

Roulette-Wheel selection

Individual order	1	2	3	4	5	6	7	8	9	10
Objective function	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2
Cumulative sum	2.0	3.8	5.4	6.8	8	9.0	9.8	10.4	10.8	11.0
Selection probability	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02

Random number (between 0 and 11.0)	3.1	0.5	7.1	10.6	6.1
Individual selected	2	1	5	8	4

SELECTION

Stochastic universal selection

This procedure is similar to the roulette-wheel selection, but now pointers equally separated replace the random numbers. The first pointer is obtained from a number generated randomly ranging between 0 and the prescribed distance, $D_{pointer}$, where:

$$D_{pointer} = \frac{\sum_{i=1}^{N} FO_i}{NS}$$

Individual order	1	2	3	4	5	6	7	8	9	10
Objective function	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2
Cumulative sum	2.0	3.8	5.4	6.8	8	9.0	9.8	10.4	10.8	11.0

The distance between pointers is 1.833 (11/6) and the number randomly generated is 0.516 (*NS* is the number of individuals to be selected)

Pointer	0.516	2.349	4.182	6.015	7.848	9.681
Individual selected	1	2	3	4	5	7

SELECTION

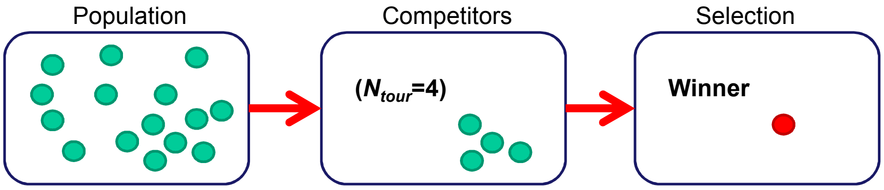
Tournament selection

- This technique consists on the random selection of a fixed number of individuals (N_{tour}) from the entire population.
- The best is picked up from this sub-group and used as parent.
- The process is repeated whenever necessary to produce a new individual.
- The value of N_{tour} can be defined as a percentage of the total population (N) and can vary between 2 and N.

SELECTION

Tournament selection

- Select N_{tour} random individuals from the population
- Take the best (eventually with some probability)
  - N_{tour} is called the size of the tournament



Population

Competitors

(N_{tour}=4)

Selection

Winner

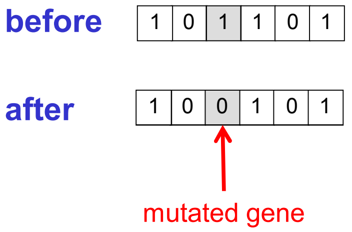
MUTATION

Mutation Operators

- At least one mutation operator should be used in order to allow every part of the search space to be reached
- The size of mutation is important and should be controllable
- Mutation should produce valid chromosomes
- Heuristics for performing mutation can be used

MUTATION

Mutation for Discrete Representation



before	1	0	1	1	0	1
after	1	0	0	1	0	1

Mutation is usually performed with a low probability (p_m) for each gene.

MUTATION

Mutation Procedure

Selection of:

	5	4	3	2	1	0	position	Decimal value
Chromosome	1	0	1	1	0	1		45

Random numbers:

position = 3

n. between 0 and 1 = 0.05 **Smaller than the mutation ratio = 0.08**

integer n. between 0 e 1 =

New Chromosome	0	1	0	0	1	0	1	37
	or							
	1	1	0	1	1	0	1	45

Mutation for real representation

Perturbation by adding some random noise

Often, a Gaussian/normal distribution N(0,σ) is used, where

- 0 is the mean value
- σ is the standard deviation

Then, $x'_i = x_i + N(0,\sigma_i)$, made for each parameter.

Mutation for real representation

This type of mutation is applied for real representation with a lower probability. A polynomial distribution of probabilities is used, being the average the actual value and the variance the index ***n***. The following perturbation factor is applied:

$$\delta = \frac{c-p}{\Delta_{\max}}$$

$\Delta_{\max}$ is a fixed quantity representing the maximum perturbation allowed, ***p*** is the parent value and ***c*** is the value to be obtained after mutation. The probability distribution ranging in the interval [-1,1] is:

$$P(\delta) = 0.5(n+1)(1-|\delta|)^n$$

MUTATION

Mutation Procedure

1- Select randomly, in the interval [0,1], a ***u*** value;

2- Calculate the perturbation factor δ corresponding to ***u*** (based on the mentioned probability distribution) using the following equation:

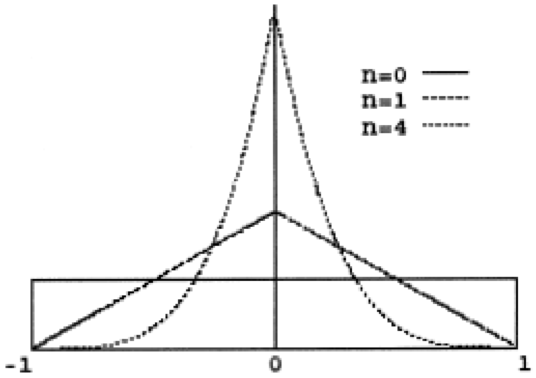
$$\delta = \begin{cases} (2u)^{\frac{1}{n+1}} - 1 & \Leftarrow \quad u \leq 0.5 \\ 1 - [2(1-u)]^{\frac{1}{n+1}} & \Leftarrow \quad u > 0.5 \end{cases}$$

3- Then, the new individual to be obtained is calculated from:

$$c = p + \delta \, \Delta_{\max}$$

MUTATION

Probability distribution for different *n* values



Mutation Rates

- The rate of mutation is an important parameter, since it can allow a good balance between exploration an exploitation
- A possibility is to allow every gene in the chromosome to have an equal chance of mutation. Usually, this probability is fixed as $1/l$, where l is the number of genes
- Thus, it is important to define exactly what rate o mutation mean (as in the examples shown above)

Recombination Operators

It will be necessary to have one or more recombination operators, mainly due to the different representations.

Some important issues are:

- The offspring must inherit some information from **each** parent;
- The recombination operator to use depends on the chromosome representation;
- Recombination should produce valid chromosomes.

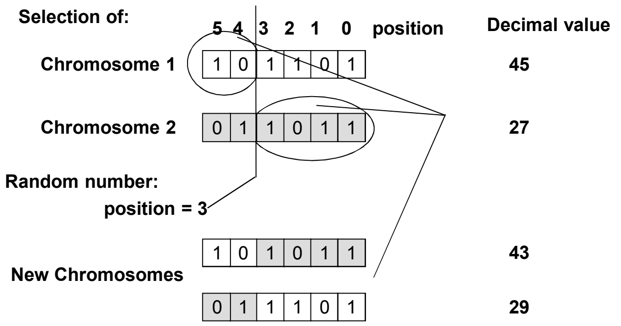
RECOMBINATION

Types of Crossover

Binary codification:
- One-point
- Two-point
- Uniform

Real codification:
- SBX
- PCX

One-point crossover



Selection of:

5 4 3 2 1 0 position **Decimal value**

Chromosome 1 1 0 1 1 0 1 **45**

Chromosome 2 0 1 1 0 1 1 **27**

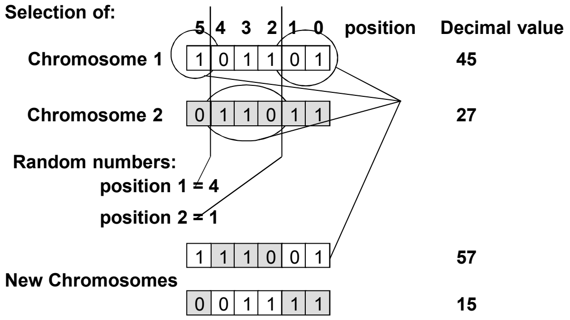
Random number:
position = 3

New Chromosomes
1 0 1 0 1 1 **43**
0 1 1 1 0 1 **29**

RECOMBINATION

Types of Crossover

Two-point crossover



Selection of:

5 4 3 2 1 0 position **Decimal value**

Chromosome 1 1 0 1 1 0 1 **45**

Chromosome 2 0 1 1 0 1 1 **27**

Random numbers:
position 1 = 4
position 2 = 1

New Chromosomes
1 1 1 0 0 1 **57**
0 0 1 1 1 1 **15**

RECOMBINATION

Types of Crossover

Uniform crossover

Selection of:

	5	4	3	2	1	0	position	Decimal value
Chromosome 1	1	0	1	1	0	1		**45**
Chromosome 2	0	1	1	0	1	1		**27**
Random numbers:								
mask 1 =	0	1	1	0	0	0		
mask 2 =	1	0	0	1	1	1		
New Chromosomes	0	0	1	0	1	1		**11**
	1	1	1	1	0	1		**61**

RECOMBINATION

Types of Crossover

SBX – Simulated Binary Crossover

1- Select randomly, in the interval [0,1], a ***u*** value;

2- Calculate β_q using the following equation:

$$\beta_q = \begin{cases} (2u)^{\frac{1}{\eta+1}}, & \text{if } u \leq 0.5 \\ \dfrac{1}{(2(1-1))^{\frac{1}{\eta+1}}}, & otherwise \end{cases}$$

η – distribution index (good value is 2)

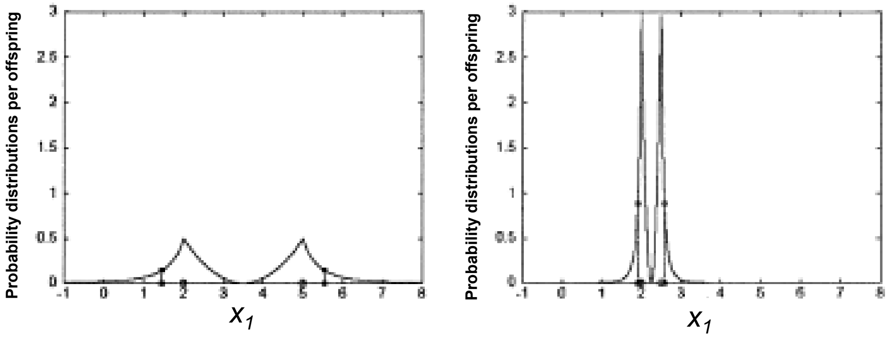
3- Calculate the offspring's from:

$$x_i^{(1,t+1)} = 0.5\left[\left(1+\beta_q\right)x_i^{(1,t)} + \left(1-\beta_q\right)x_i^{(2,t)}\right]$$
$$x_i^{(2,t+1)} = 0.5\left[\left(1-\beta_q\right)x_i^{(1,t)} + \left(1+\beta_q\right)x_i^{(2,t)}\right]$$

RECOMBINATION

Types of Crossover

SBX – Simulated Binary Crossover



Probability of distributions of the offspring's calculated by SBX

RECOMBINATION

Types of Crossover

PCX– Parent-Centric Crossover

1. Random selection of 3 solutions, x_1, x_2, x_3 (parents);
2. An average vector is calculated as:

$$g = \left(\frac{x_1^1 + x_2^1 + x_3^1}{3}, \frac{x_1^2 + x_2^2 + x_3^2}{3}, \frac{x_1^3 + x_2^3 + x_3^3}{3} \right)$$

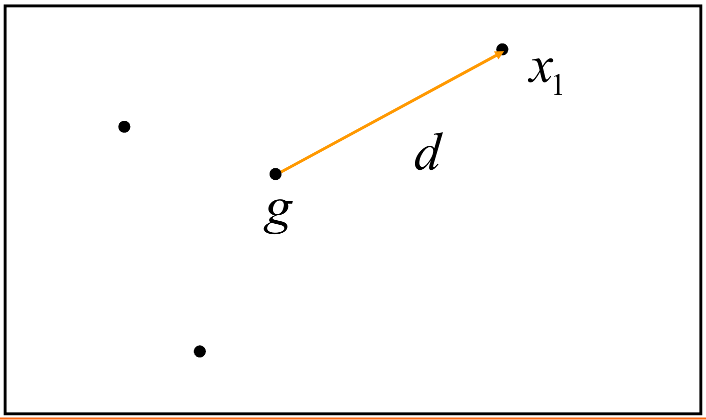
3. One of the parents is selected randomly. For this example, without loss of generality, x_1 is selected;

RECOMBINATION

Types of Crossover

PCX– Parent-Centric Crossover

4. A direction vector is calculated as: $d = x_1 - g$



RECOMBINATION

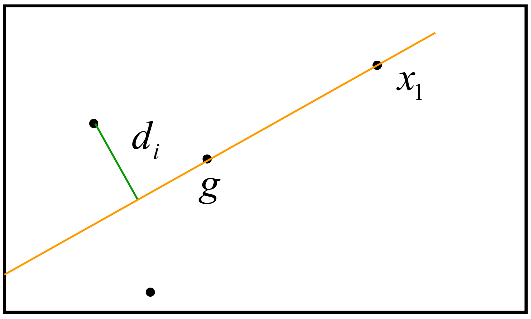
Types of Crossover

PCX– Parent-Centric Crossover

5. The distance between the solution ***i*** and the line passing at ***x***$_1$ and has direction ***d*** is determined:

$$d_i = \frac{\sqrt{\left\|x_i - x_1\right\|^2 \left\|d\right\|^2 - \langle x_i - x_1, d\rangle^2}}{\left\|d\right\|}$$

with, $i \in \{2, 3\}$



RECOMBINATION

Types of Crossover

PCX– Parent-Centric Crossover

6. The average of the distances calculted in 5 is determined:

$$\overline{D} = \frac{\sum_{i=1, i \neq p}^{3} d_i}{2}$$

7. A random vector $w = (w_1,\ w_2,\ w_3)$, where each component has a distribution $N(0,\ \overline{D}.\sigma_\delta)$ is determined:

$$w_i = \left(\sqrt{-2\ln(x_1)}.\cos(2\pi\ x_2)\right)\overline{D}.\sigma_\delta$$

where, x_1 and x_2 are random numbers ranging in the interval [0,1]

$\sigma_\delta = 0.01$

RECOMBINATION

Types of Crossover

PCX– Parent-Centric Crossover

8. Using the process of Gram-Schmidt a vector ***u***, orthogonal to ***d***, is determined:

$$u = w - \frac{\langle w\ ,\ d \rangle}{\langle d\ ,\ d \rangle} d$$

9. A random number z_λ with distribution $N(0,\ \sigma_\lambda)$ is determined:

$$z_\lambda = \left(\sqrt{-2\ln(x_1)}.\cos(2\pi\ x_2)\right)\sigma_\lambda$$

where, x_1 and x_2 are random numbers ranging in the interval [0,1]

RECOMBINATION

Types of Crossover

PCX– Parent-Centric Crossover

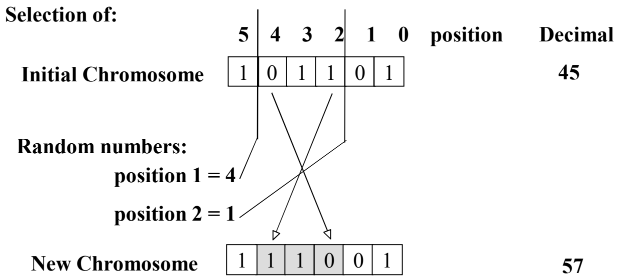
10. A new individual is generated: $y = x_1 + z_\lambda d + u$

Graphical interpretation of the PCX operator

INVERSION

Inversion Procedure

- Inversion produces the interchange of information inside one chromosome.
- Two points of the chromosome are randomly selected and their genes are inverted:



Selection of:

	5	4	3	2	1	0	position	Decimal
Initial Chromosome	1	0	1	1	0	1		45
New Chromosome	1	1	1	0	0	1		57

Random numbers:
position 1 = 4
position 2 = 1

Population Strategy

The selection pressure can be strongly affected by the way the population is deal with:

- Decision about which individuals are to be eliminated
- Decision about the population size (fixed or variable)
- The possibility of allowing duplicates in the population
- The existence of elitism (in order to keep the best solutions)
- Decision about if the whole population is replaced at once (generational) or one individual at a time (steady-state),
- Decision about the existence or not of comparisons between the offspring's and the offspring's and parents.

Replacement Strategy

- **Depending on the population strategy used, could be necessary the selection of individuals to eliminate**
- **Some deterministic replacement strategies can be used, for example, *replace worst* or *replace oldest*.**

Recombination *versus* Mutation

- **Recombination**
  - The changes depend on the entire population
  - The effects decrease with the convergence
  - It is an **exploitation** operator
- **Mutation**
  - Is required to escape local optima
  - Depends strongly from causality
  - It is an **exploration** operator

Stop criterion

- Limit on CPU resources:

  Maximum number of fitness evaluations

- Strategy based on convergence:

  After some generations without improvement

Algorithm performance

Any conclusion can be draw from a single run:

- It is necessary the use statistical measures (averages, medians)
- This implies the algorithm must be run a sufficient number of independent runs

---

Exploration *versus* Exploitation

Exploration of unknown regions

- If too much exporation is performed (which can implies random search) no convergence can be achieved

Exploitation of specific ”good” regions of the search space

- If too much expoitation is made this can implies only local search, thus convergence to a local optimum

OTHER METHODS

- Evolutionary Strategies
- Genetic Programming
- ANT Colony Algorithms

OTHER METHODS

Evolutionary Strategies

OTHER METHODS – *Evolutionary Strategies*

Historical Perspective

- The Evolutionary Strategies emerge in the end of 60´s in Germany, but was in the following decade, after the publication of Rechenberg and Schwefel works, that started to be known.
- The Rechenberg group applied several Evolutionary Strategies in optimization problems in the field of Engineering.

OTHER METHODS – *Evolutionary Strategies*

General Characteristics

- Start the search from a population of potential solutions randomly generated (if known an initial approximation can be used);
- Work, during the successive generations, with a population of potential solutions, instead of a single solution;
- Don’t use any information concerning the first and/or second derivatives of the objective function and/or restrictions;
- Do not require any condition relative to the continuity and convexity of the search space;
- Can use mechanisms that allow finding multiple local optima in a single run;
- Guide the search based on probabilistic mechanisms and/or rules.

OTHER METHODS – Evolutionary Strategies

General Characteristics

Conditions to use evolutionary strategies as optimization technique:

- Non-convex problems;
- The existence of several local optima;
- In the case of non-differentiability;
- They are efficient in identifying promising search regions.

OTHER METHODS – Evolutionary Strategies

General Characteristics

Nomenclature:

- Generations -> iterations;
- Individuals or members = points;
- Individuals are vectors of real numbers;
- Population of parents = μ individuals;
- Population of offspring's = λ individuals;
- Recombination = ρ individuals;
- Selection deterministic (the **+** and **,** selection);
- Gaussian mutation with adaptation: Isotropic, Non-Isotropic and Non-Isotropic with rotation.

OTHER METHODS – *Evolutionary Strategies*

General Characteristics

Examples:

EE-(1 + 1)

EE-($\mu + \lambda$) and EE-(μ , λ)

EE-($\mu/\rho + \lambda$) and EE-($\mu/\rho , \lambda$)

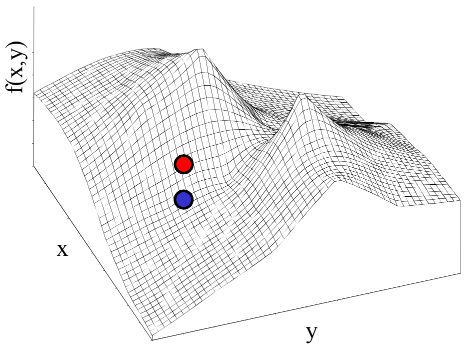
OTHER METHODS – *Evolutionary Strategies*

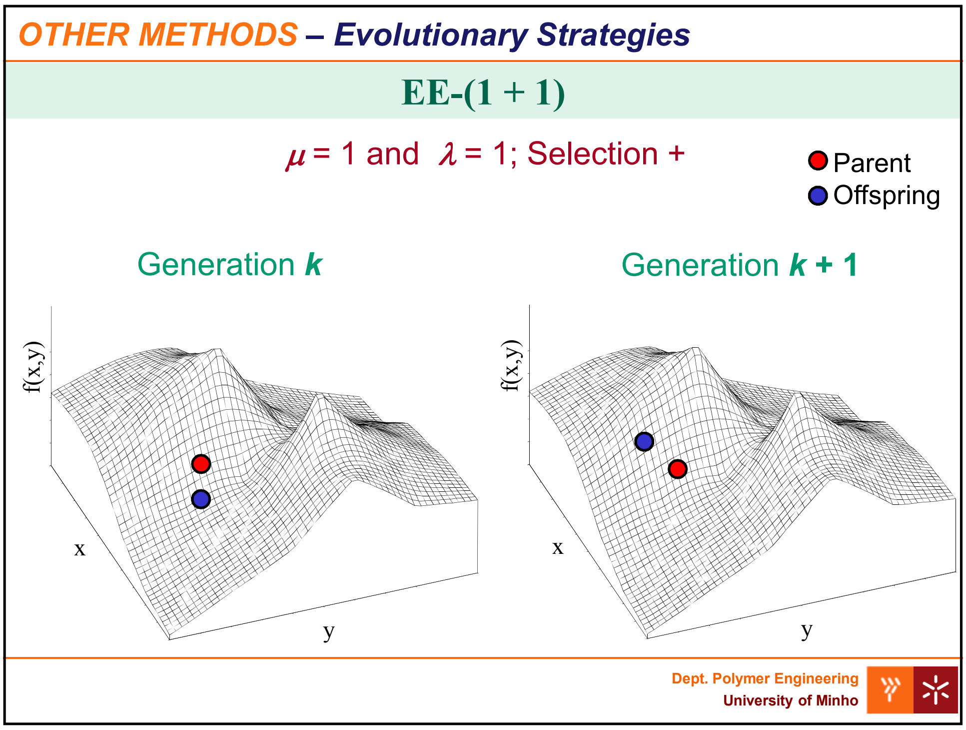
EE-(1 + 1)

$\mu = 1$ and $\lambda = 1$; Selection +

● Parent
● Offspring

Generation ***k***





OTHER METHODS – *Evolutionary Strategies*

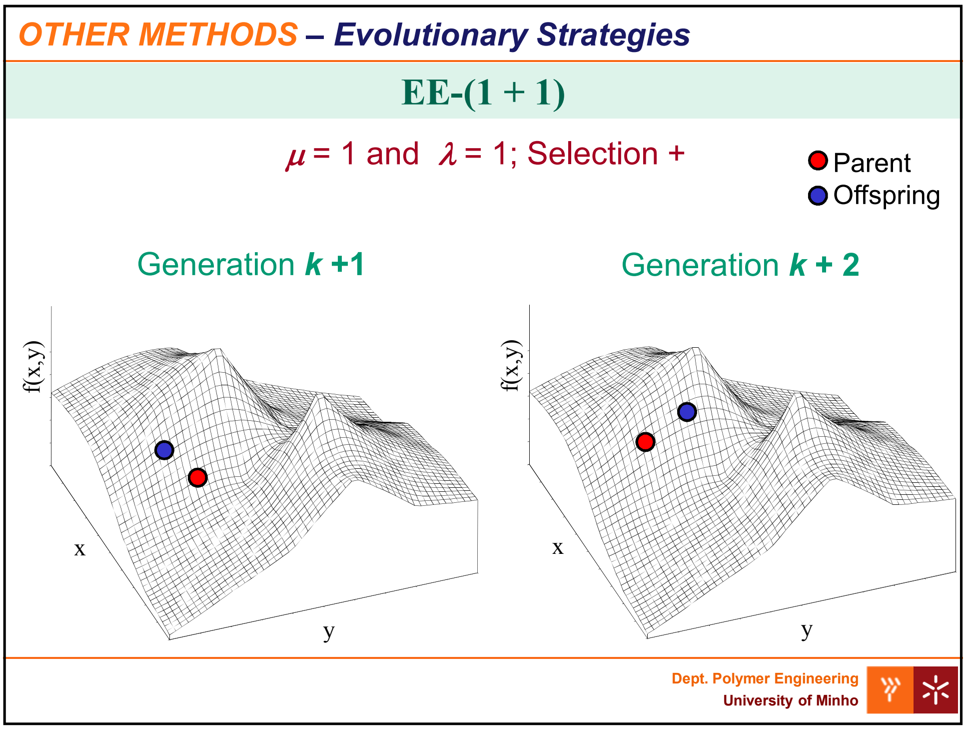
EE-(1 + 1)

μ = 1 and λ = 1; Selection +

🔴 Parent
🔵 Offspring

Generation ***k***

Generation ***k* + 1**



OTHER METHODS – *Evolutionary Strategies*

EE-(1 + 1)

μ = 1 and λ = 1; Selection +

🔴 Parent
🔵 Offspring

Generation ***k* +1**

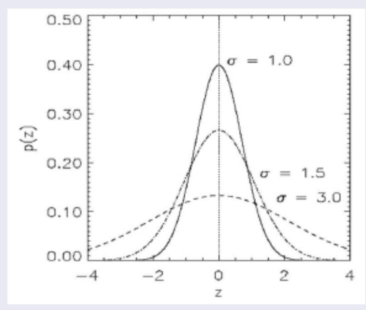
Generation ***k* + 2**

OTHER METHODS – Evolutionary Strategies

EE-(1 + 1)

Normal distribution: N(μ,σ_i)

If $\mu = 0$ then N(0,σ_i)



OTHER METHODS – Evolutionary Strategies

EE-(1 + 1)

- Initial approximation (starting point): x_0
- Initial value for the standard deviation: σ

$$\sigma = \frac{\Delta x}{\sqrt{n}}$$

Where, Δx is an approximate measure of the expected distance between the initial solution and the optimum and n is the problem dimension.

- **Question:**

How to select the value of σ during the search process, in order to obtain improvements?

➡ Rule of “1/5 of successes” of Rechenberg (1973).

OTHER METHODS – *Evolutionary Strategies*

EE-(1 + 1)

1/5th success rule:

applied periodically to all the Δt iterations

$$\sigma^{(k+1)} = \begin{cases} c_{inc}\sigma^{k} & \Leftarrow P_s(\Delta t) > 1/5 \\ c_{dec}\sigma^{k} & \Leftarrow P_s(\Delta t) < 1/5 \\ \sigma^{k} & \Leftarrow P_s(\Delta t) = 1/5 \end{cases}$$

$P_s(\Delta t)$ is the percentage of successes during the last Δt iterations

$c_{dec}<1$ and $c_{inc}<1$, are the decrease and increase coefficients of the standard deviation.

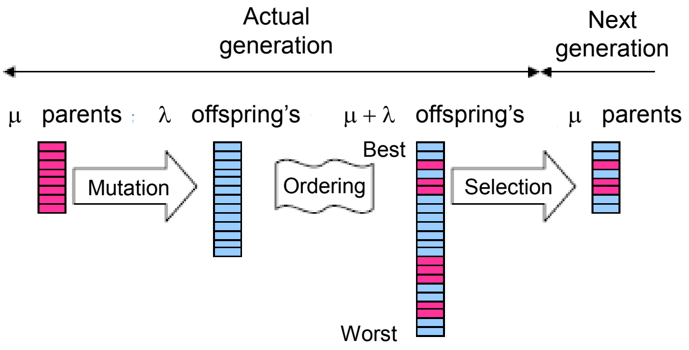
NOTES:

$c_{dec}<=0.85$, $c_{inc}<=1/0.85$ and $\Delta t=10n$

$$\sigma_1^k = \sigma_2^k = \ldots = \sigma_n^k = \sigma^k$$

OTHER METHODS – *Evolutionary Strategies*

EE-($\mu + \lambda$)



Actual generation

Next generation

μ parents

λ offspring's

μ + λ offspring's

μ parents

Mutation

Ordering

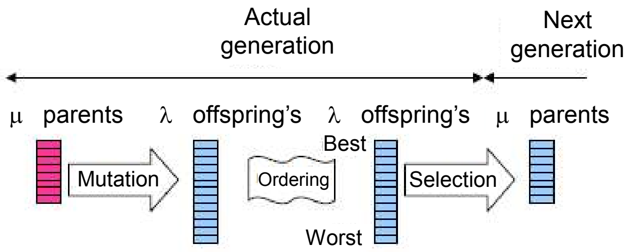
Selection

Best

Worst

OTHER METHODS – *Evolutionary Strategies*

EE-(μ , λ)



Actual generation

Next generation

μ parents

λ offspring's

λ offspring's

μ parents

Mutation

Ordering

Best

Worst

Selection

OTHER METHODS

Genetic Programming

Origins

- ☐ Genetic Programming (GP) is the newest approach inside the area of Evolutionary Computation (EC).
- ☐ It was created by John Koza (1992).
- ☐ The origins of Genetic Programming were the Genetic Algorithms.
- ☐ John Koza was a PhD student of John Holland, the father of GAs.

Characteristics

The main difference between GP and GAs is related with representation of the individuals:

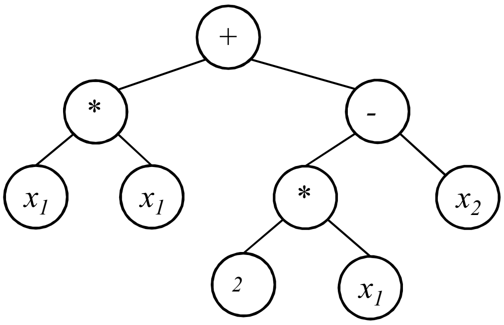
- **GA representation** – fixed length numerical strings
- **GP representation** – variable length structures containing whatever ingredients are needed to solve the problem

OTHER METHODS

Representation

The most popular is tree-based representation

$f(x_1, x_2) = x_1^2 + 2x_1 - x_2$



OTHER METHODS

Initialization

The initial trees must be generated without exceeding a pre-defined depth (typically 6).

Three different initialization methods can be used.

They must start from the root of the tree, and nodes are added until the leaves:

- Grow - maximum depth cannot be exceeded
- Full - tree must be full (all branches must reach maximum depth)
- Ramped - half of the trees are initialized with Grow and the other half with Full

OTHER METHODS

Selection

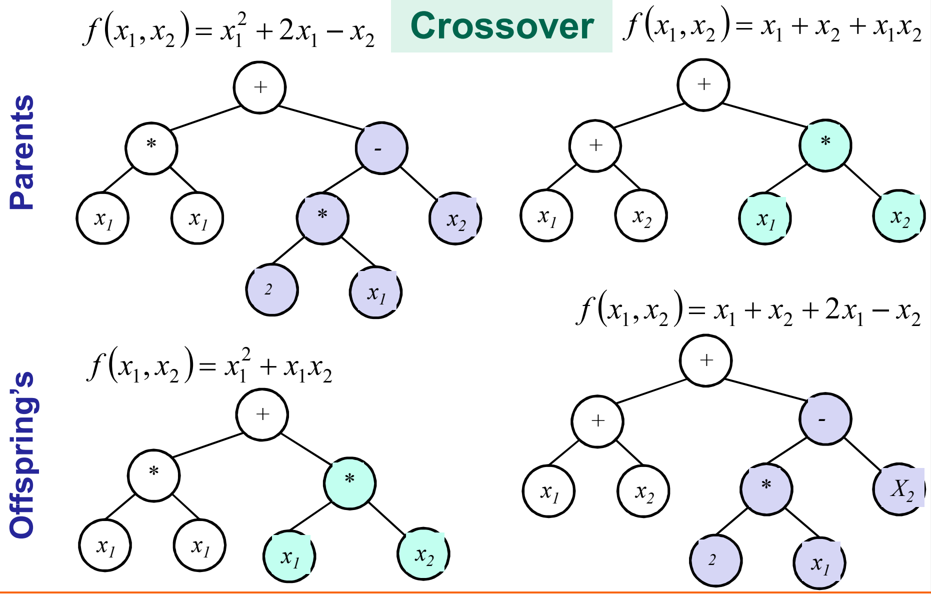
The selective pressure used is usually determined by the method used:

- **Roulette** selection impose a high selection pressure,
- **Tournament** selection allow a higher control of selection pressure (by controlling the size of the tournaments).

Multi-objective optimization can be performed using some tournament methods (like Double Tournament).

OTHER METHODS

Crossover



$f(x_1,x_2)=x_1^2+2x_1-x_2$

$f(x_1,x_2)=x_1+x_2+x_1x_2$

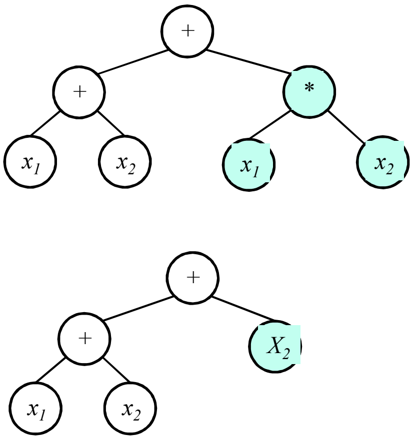
$f(x_1,x_2)=x_1+x_2+2x_1-x_2$

$f(x_1,x_2)=x_1^2+x_1x_2$

OTHER METHODS

Mutation

Parent



$$f(x_1,x_2)=x_1+x_2+x_1x_2$$

Offspring

$$f(x_1,x_2)=x_1+x_2+x_2$$

OTHER METHODS

ANT Colony Algorithms

Ant Colony Optimization

- ➢ Ant Colony Optimization (ACO) studies artificial systems that take inspiration from the behaviour of real ant colonies.
- ➢ Marco Dorigo introduced the first ACO system in 1992, named Ant System (AS).
- ➢ AS was initially applied to the travelling salesman and to the quadratic assignment problems.
- ➢ The **Ant Colony Optimization metaheuristic** was defined by Dorigo, Di Caro and Gambardella in 1999.

Historical Perspective

• Ant System (AS)	Dorigo et al 1991
• Elitist AS (EAS)	Dorigo et al 1992
• Ant-Q	Gambardella & Dorigo 1995
• Ant Colony System (ACS)	Dorigo & Gambardella 1996
• Max-Min AS (MMAS)	Stützle & Hoos 1996
• Rank-Based AS (ASRank/RBAS)	Bullnheimer et al 1997
• ANTS	Maniezzo 1999
• Best-Worst AS (BWAS)	Cordón et al 2000
• Hyper-Cube AS (HCAS)	Blum et al 2001
• …	

Applications

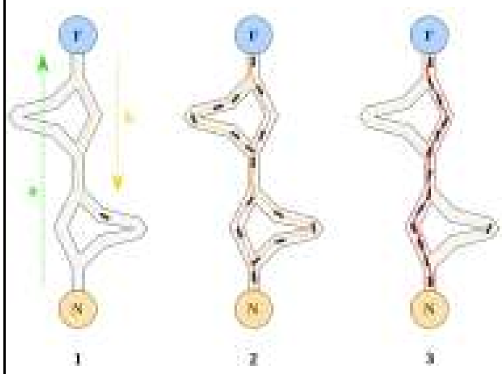
- **Scheduling :** Job-shop scheduling problem, Permutation flow shop problem, Single machine total tardiness problem, etc…
- **Vehicle routing:** Capacitated vehicle routing problem, Multi-depot vehicle routing problem, etc…
- **Assignment:** Quadratic assignment problem, Generalized assignment problem, Frequency assignment problem, etc…
- **Set problem:** Set covering problem, Set partition problem, Multiple knapsack problem, etc…
- **Others:** Classification, Connection-oriented network routing, Connectionless network routing, Image processing, Protein Folding, etc…

Stigmergy

- Stigmergy is the phenomena that explains the collective behaviour's exhibited by the ant/termite colonies (nest building, division of labour, structure formation, cooperative transport)
- Stigmergy can be seen/defined as the process of a specific agent to exchange information indirectly with other agents (usually of the same specie). In the case of the ants, this is made by depositing pheromone.
- The *Ant Colony Optimization metaheuristic (ACO)* [Dorigo & Di Caro, 1999] is the most popular, general, and effective SI framework based on these principles.

Working Principles

The idea was originated by the observation of the exploitation of food resources among ants. The ants' have collectively been able to find the shortest path between a food source and the nest (besides their individually limited cognitive abilities):



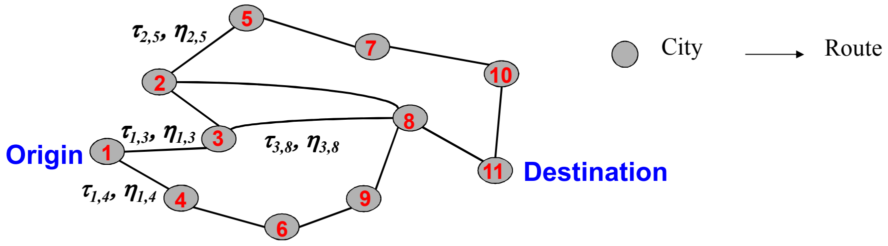
1. The first ant finds the food source (F), via any way (a), then returns to the nest (N), leaving behind a trail pheromone (b)
2. Ants indiscriminately follow four possible ways, but the strengthening of the runway makes it more attractive as the shortest route.
3. Ants take the shortest route, long portions of other ways lose their trail pheromones.

Steps of an ACO

1. Represent the problem using sets of components and transitions, or a graph where the ants will operate.
2. Define what would be the pheromone trails (array) for the problem.
3. Define what would be the heuristic information (array), associated with the problem, which affects the decisions of the ants.
4. Whenever possible, implement an efficient local search algorithm related to the problem being solve.
5. Choose/adapt one of variants of existing ACO.
6. From values previously used in similar applications, adjust the parameters of the ACO for the problem.

OTHER METHODS

Representation



Matrix for the
Heuristic variable
(e.g., distance travelled)

$\eta_{1,1}$	$\eta_{1,2}$	…	$\eta_{1,n}$
$\eta_{2,1}$			
…			
$\eta_{n,1}$			

Matrix for the
Pheromone

$\tau_{1,1}$	$\tau_{1,2}$	…	$\tau_{1,n}$
$\tau_{2,1}$			
…			
$\tau_{n,1}$			

OTHER METHODS

Algorithm

1. Initialize parameters
2. Initialize heuristic matrix
3. Initialize pheromone matrix
4. **WHILE** stop condition not satisfied
   a. Build/Evaluate solutions
   b. Apply local search (optional)
   c. Actualize pheromone matrix
5. **End while**
6. **Return** the best solution found

ANT-routing Table

At each node (*i*), the values of heuristic and pheromone matrixes must be combined in such a way that give a relative weight in each node. This enables the possibility of assigning a precise goodness value to each locally available next step (*j*):

$$A_i(j) = f_\tau(\tau_i, j) \circ f_\eta(\eta_i, j)$$

Examples:

$$A_i(j) = \tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}$$

$$A_i(j) = \alpha\tau_{i,j} + (1-\alpha)\eta_{i,j}$$

Decision Policy

- Each ant can be seen as an *autonomous agent that constructs a path (i.e.,* proposes a solution to the problem).
- Ants do not need synchronization, thus might be one or more ants *concurrently active* at the same time.
- Next steps are selected using a *stochastic decision policy*, such as for example:

$$p_{i,j} = \frac{A_i(j)}{\sum_{k \in N(i)} A_i(k)}$$

OTHER METHODS

Pheromone Update

1- Pheromone evaporation:

$$\tau_{i,j} \leftarrow (1-\rho)\tau_{i,j} \;\; \forall (i,j) \in N(i)$$

2- Pheromone deposition:

$$\tau_{i,j} \leftarrow \tau_{i,j} + \sum_{k=1}^{n} \Delta\tau_{i,j}^{k} \;\; \forall (i,j) \in N(i)$$

Where, $0 < \rho \leq 1$ is the evaporation ration and $\Delta\tau_{i,j}^{k} = 1/C^{k}$ is the quantity of pheromone deposited by ant k in the edges visited.

OTHER METHODS

Design of an ANT Algorithm

- Representation of the problem
- Pheromone model
- Heuristic variables
- Ant-routing table
- Stochastic decision policy
- Solution evaluation
- Policies for pheromone updating
- Scheduling of the ants
- Pheromone initialization, constants, . .