



Programa de doctorado interuniversitario
en Tecnologías de la Información

Curso: Técnicas de Computación Flexible

<http://sci2s.ugr.es/docencia/index.php>

Genetic Algorithms: Basic notions and some advanced topics

Francisco Herrera

Grupo de Investigación

“Soft Computing y Sistemas de Información Inteligentes”

Dpto. Ciencias de la Computación e I.A.

Universidad de Granada

18071 – ESPAÑA

herrera@decsai.ugr.es

<http://sci2s.ugr.es>



DECSAI
Universidad de Granada

MATERIAL DEL CURSO

Enlace: <http://sci2s.ugr.es>

Members

Research Lines

Advised PhD

Publications

Highly Cited Papers

Review, Taxonomy & Position Papers

Edited Books & Special Issues

Editorial Boards

Citation Reports h & g Indexes

Conference Activities

Projects

KEEL

SECABA

Collaborations & Visitors

Research Lines Bibliography

Links of Interest

Seminars

Teaching

<http://sci2s.ugr.es>

Research Group
"Soft Computing and Intelligent Information Systems"

The research group is organized in seven different laboratories:

 Evolutionary & Fuzzy DATA Mining & Intelligent Systems	 Bioinformatics Mining, Modeling, Annotating, Predicting	 Decision Making	 Image Registration Applying Metaheuristics	 Genetic Algorithms & Biologically Inspired Computation	 SECABA (Quality Evaluation & Information Retrieval)	 Distributed Computational Intelligence and Time Series
--	---	---------------------	--	--	---	--

© Copyright 2003-2007, **SCI²S** (Soft Computing and Intelligent Information Systems)
[About the Webmaster Team](#)

MATERIAL DEL CURSO

Enlace: <http://sci2s.ugr.es/docencia/index.php>

Other Post-graduate Courses



[Programa de Doctorado - Ingeniería de Estructuras - Curso: Optimización y Computación Inteligente \(Granada\)](#)

[Programa de Doctorado Interuniversitario en Tecnologías de la Información - Curso: Técnicas de Computación Flexible \(Santiago de Compostela\)](#)

[Future Directions in Soft Computing \(Mieres - Asturias\) - Genetic Algorithms, Basic Notions and some Advanced Topics](#)

[Dottorato di Ricerca in Ingegneria dell'Informazione \(Pisa - Italy\) - Data Mining and Soft Computing](#)

MATERIAL DEL CURSO

Enlace: http://sci2s.ugr.es/docencia/asignatura.php?id_asignatura=09

The screenshot shows a Mozilla Firefox browser window with the URL http://sci2s.ugr.es/docencia/asignatura.php?id_asignatura=09. The page features a navigation menu on the left with links such as 'Presentation', 'Members', 'Research Lines', 'Advised PhD', 'Publications', 'Highly Cited Papers', 'Review, Taxonomy & Position Papers', 'Edited Books & Special Issues', 'Editorial Boards', 'Citation Reports h & g Indexes', and 'Conference Activities'. The main content area includes the DECSAI logo, a banner for 'Doctorado Santiago de Compostela', and the course title 'Programa de Doctorado Interuniversitario en Tecnologías de la Información' and 'Curso: Técnicas de Computación Flexible'. Below this, it lists the instructor 'Francisco Herrera (Dpto. de Ciencias de la Computación e I.A.)' and a 'Documentación' section with four sessions:

- **Sesión 1:** Genetic Algorithms. Basic concepts. ([PDF, 1319 Kb](#))
- **Sesión 2:** Genetic Algorithms. Advanced concepts. ([PDF, 2390 Kb](#))
- **Sesión 3:** Genetic Fuzzy Systems: Basic Notions and Tuning Methods. ([PDF, 1829 Kb](#))
- **Sesión 4:** Genetic Fuzzy Systems. HVAC Application. ([PDF, 3504 Kb](#))

At the bottom of the page, there is a link for 'Material Complementario'.

F. Herrera - Introduction to Genetic Algorithms

Genetic Algorithms:

Basic notions and some advanced topics

SESSIONS

a. Introduction to genetic algorithms

b. Advanced topics

Multimodal problems and multiple solutions

Multiobjective genetic algorithms

Memetic algorithms

Genetic Learning

Session a.

Genetic Algorithms

1. **GENETIC ALGORITHMS. INTRODUCTION**
2. **HOW TO CONSTRUCT THEM?**
3. **ON THE USE OF GENETIC ALGORITHMS**
4. **MODELS: GENERATIONAL VERSUS STEADY STATE**
5. **APPLICATIONS**
6. **EXAMPLE: TSP**
7. **SOFTWARE AND IMPLEMENTATIONS**
8. **CONCLUDING REMARKS**

1. GENETIC ALGORITHMS. INTRODUCTION

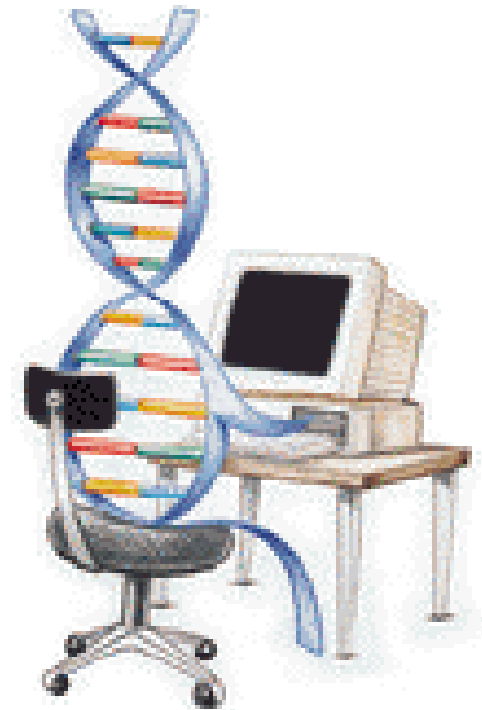
- **WHAT IS A GENETIC ALGORITHM?**
- **THE INGREDIENTS**
- **THE EVOLUTION CYCLE**
- **GENETIC ALGORITHM STRUCTURE**

What is a genetic algorithm?

Genetic algorithms

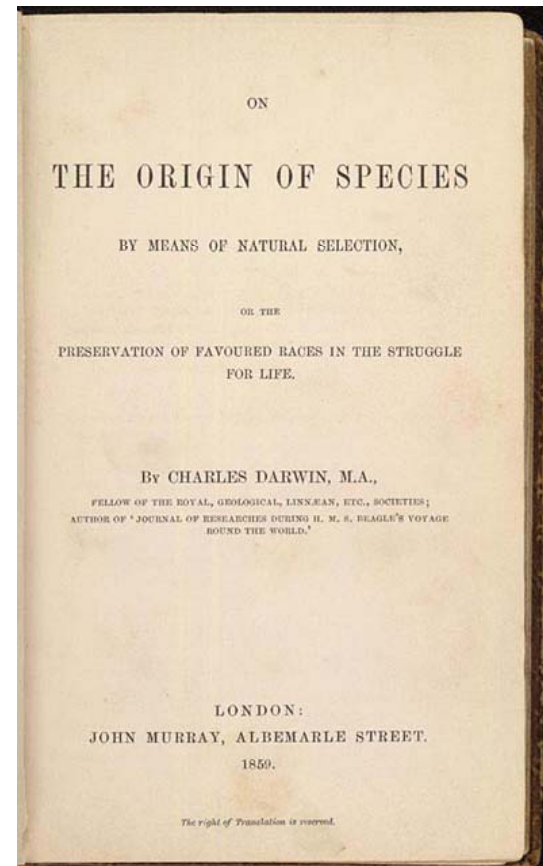
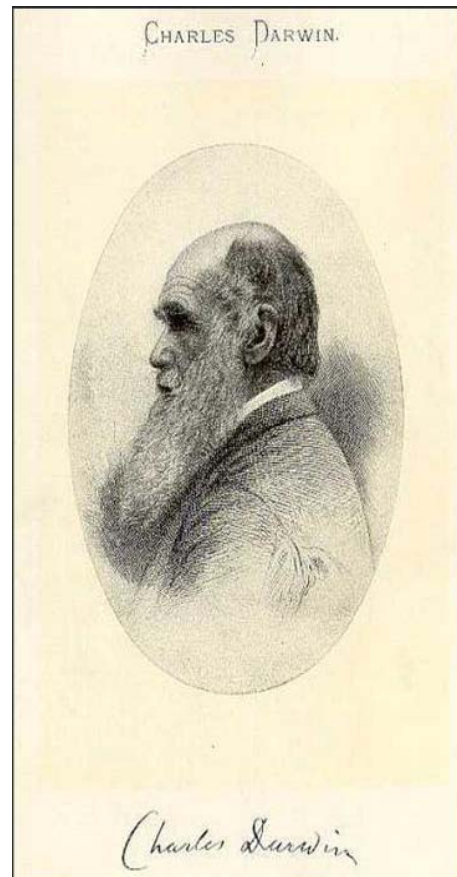
They are optimization algorithms,
search
and learning
inspired in the process of

**Natural and Genetic
Evolution**



What is a genetic algorithm?

Natural Evolution

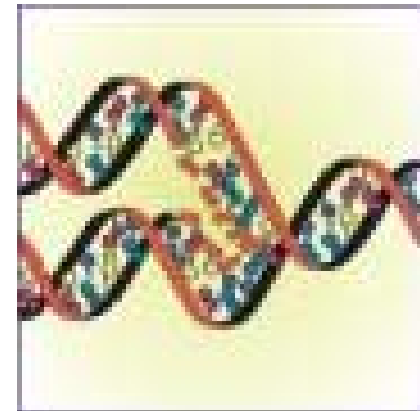


What is a genetic algorithm?

Artificial Evolution

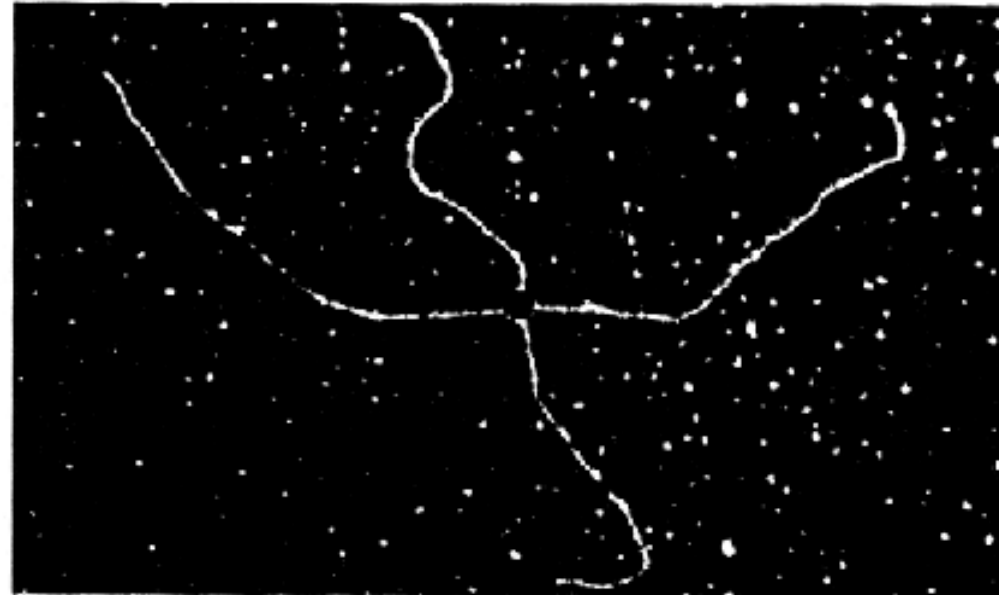
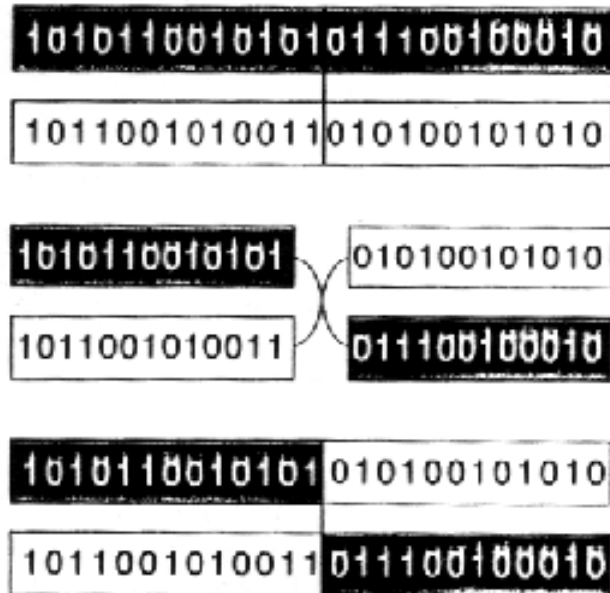
EVOLUTIONARY COMPUTATION

It is constituted by evolutionary models based on populations whose individuals represent solution to problems.



From natural to artificial evolution

Classical image (John Holland): Biological crossover



CROSSOVER is the fundamental mechanism of genetic rearrangement for both real organisms and genetic algorithms.

Chromosomes line up and then swap the portions of their genetic code beyond the crossover point.



What is a genetic algorithm?

Artificial Evolution

There are 4 classic paradigms:

Genetic Algorithms. 1975, Michigan University



John Holland
Inventor of genetic algorithms
Professor of CS and Psychology at the U. of Michigan.

Evolution Strategies 1964, Technische Universität Berlin



Inventors of Evolution Strategies



Hans-Paul Schwefel
Universität Dortmund

Ing. Ingo Rechenberg
Bionics & Evolutionstechnik
Technical University Berlin
<http://www.bionik.tu-berlin.de/>

Evolutionary Programming. 1960-1966, Florida



Lawrence J. Fogel,
Natural Selection, Inc.
Inventor of Evolutionary Programming

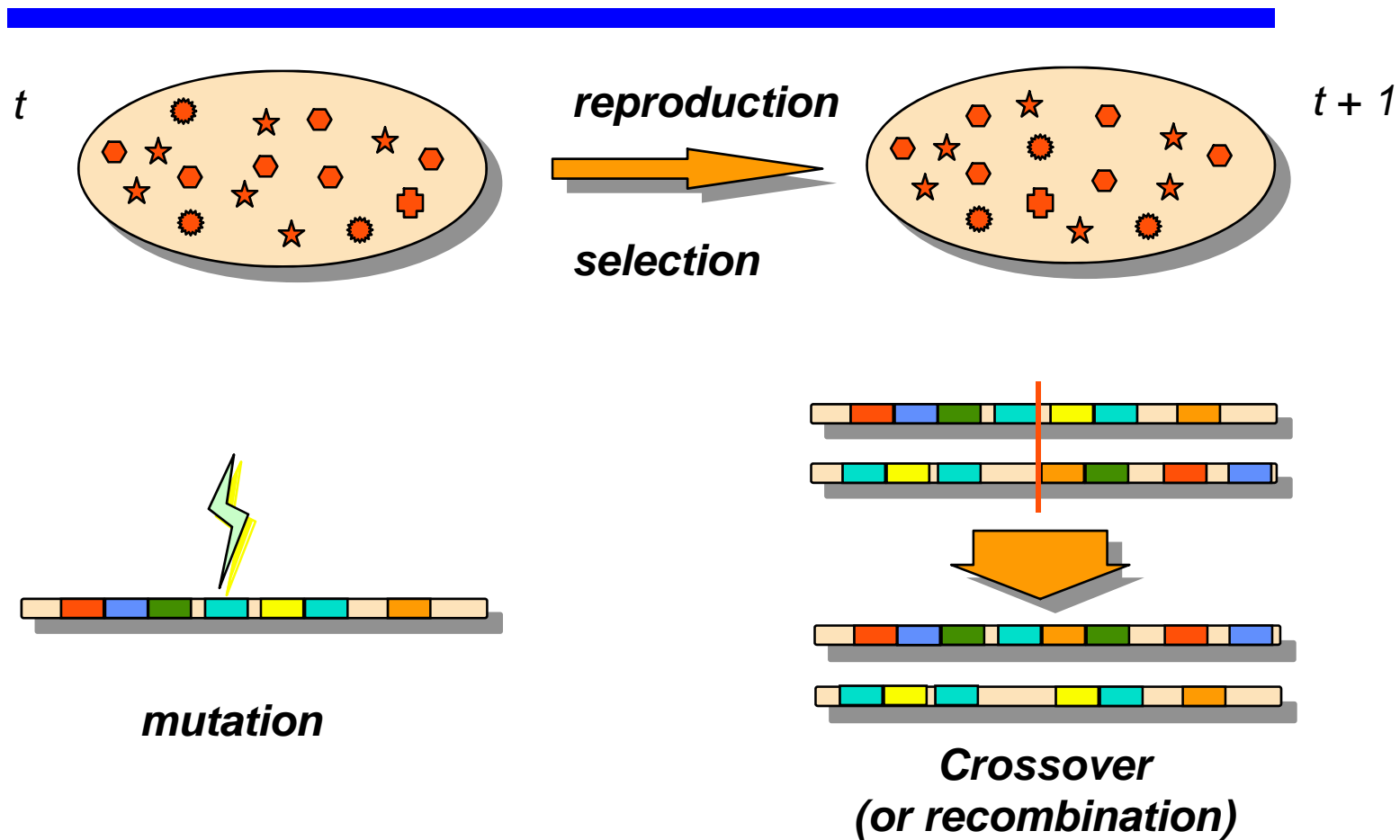
Genetic Programming. 1989, Stanford University



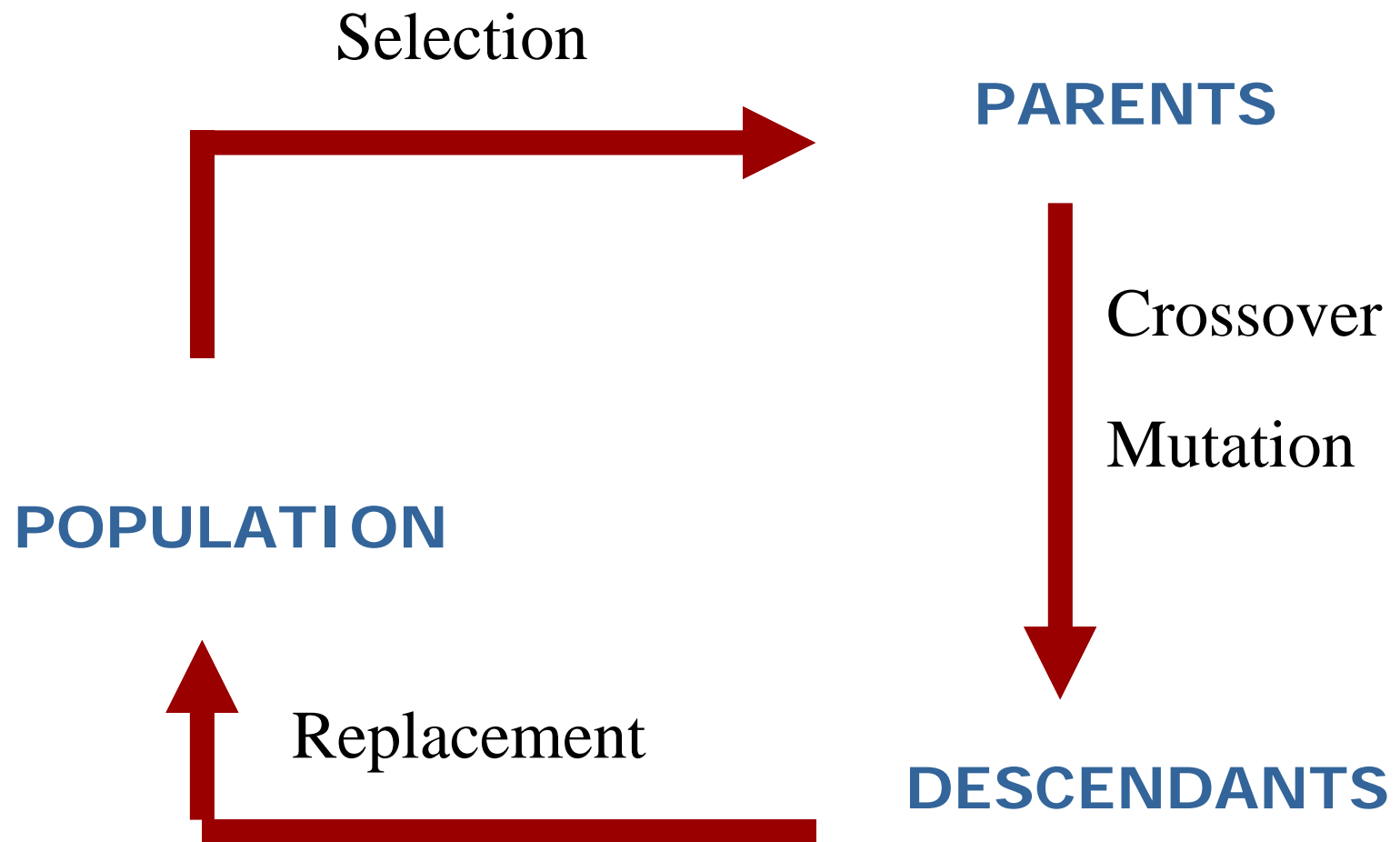
John Koza
Stanford University.
Inventor of Genetic Programming

There exist other models based on population evolution algorithms

The ingredients



The evolution cycle



Genetic Algorithm Structure

Basic Genetic Algorithms

Beginning (1)

$t = 0$

Initialization $P(t)$

evaluation $P(t)$

While (the stop condition is not verified) do

Beginning (2)

$t = t + 1$

selection $P'(t)$ from $P(t-1)$

$P''(t) \leftarrow$ crossover $P'(t)$

$P'''(t) \leftarrow$ mutation $P''(t)$

$P(t) \leftarrow$ replacement ($P(t-1), P'''(t)$)

evaluation $P(t)$

Final(2)

Final(1)

2. HOW TO CONSTRUCT A GA?

The steps for the GA construction

- Representation
- Initial population
- Fitness function (How to evaluate a GA?)
- Chromosomes selection for parents
- Design of crossover operator
- Design of mutation operator
- Chromosomes replacement
- **Stop condition**

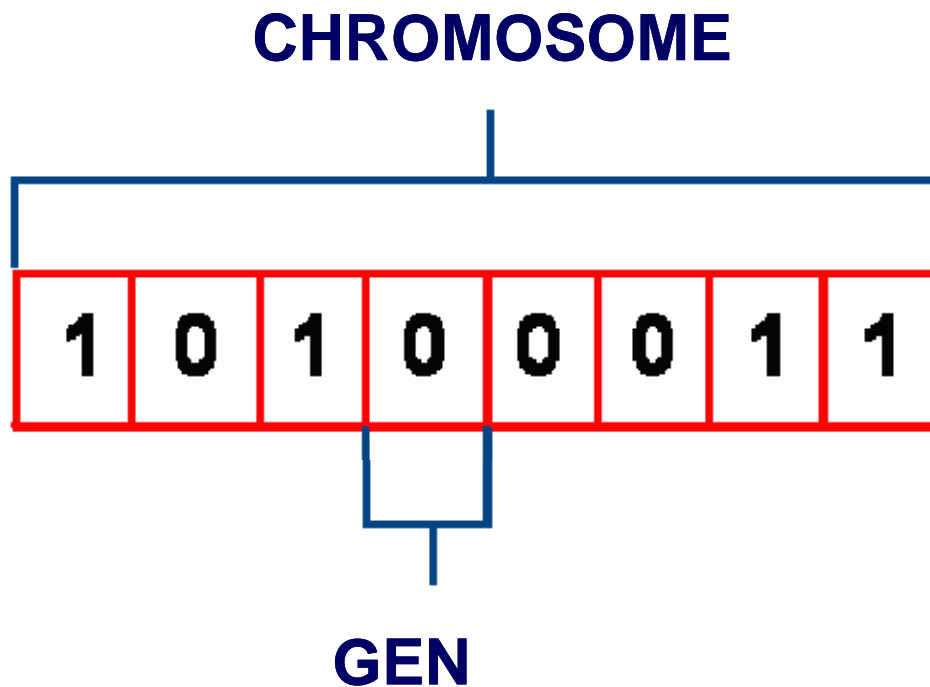
PROBLEM
DEPENDENT

ALGORITHM
COMPONENTS

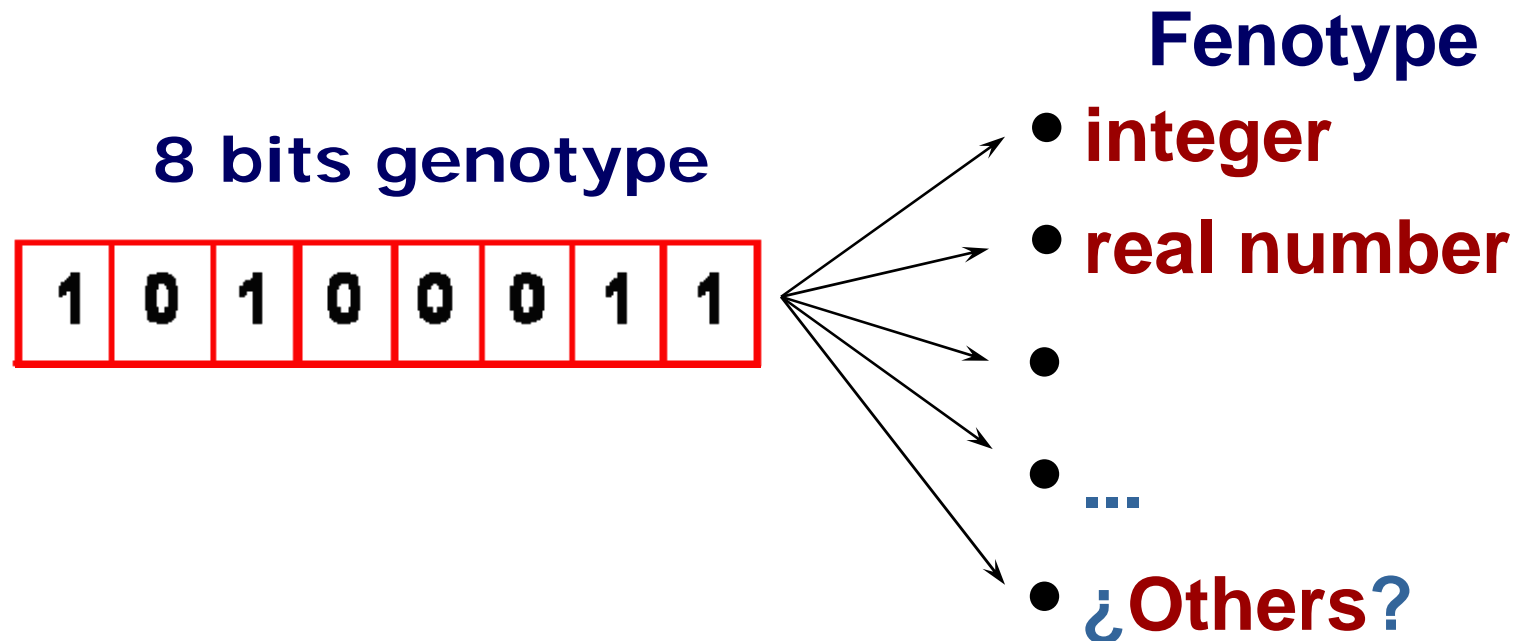
Representación

- Genotype: Coding mechanism
- Natural representation for the problem
- Genotype representation must be decided according to the evaluation and genetic the operators.

Example: Binary representation



Example: Binary representation



Example: Real coding

- The chromosome can be represented by a real valued vector:

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

- The evaluation function associates a real value to a vector:

$$f : R^n \rightarrow R$$

Example: Order representation

- The chromosomes are presented as permutations.
- Ej. Travelling salesman problem (TSP), ...
- It needs special operators for obtaining a new permutation.

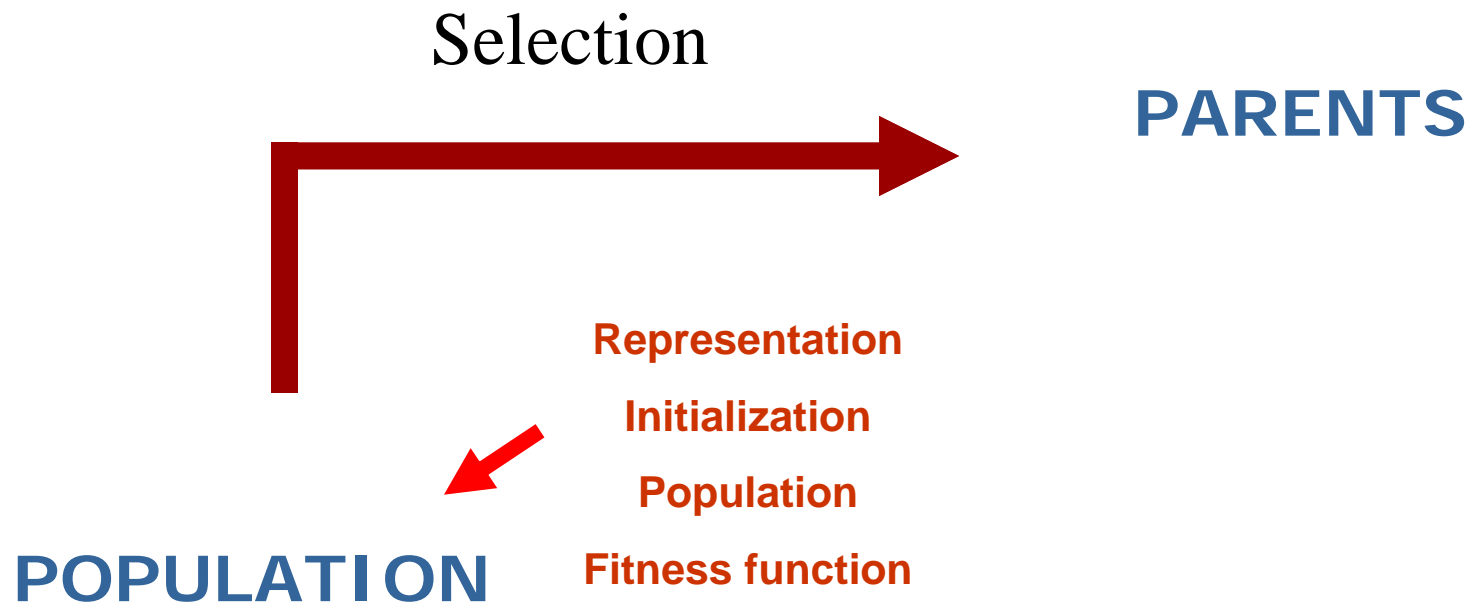
Initialization

- Uniform on the search domain ... (if possible)
 - Binary string: 0 or 1 with probability 0.5
 - Real value: uniform on the interval
- Using a heuristic for getting initial chromosomes.

Fitness function

- Step with high time cost.
- Subroutine, simulator or other external processes (ej. Robot experiment, ...)
- It is possible to use an approximation function (reducing the cost)
- Constraint problems can introduce a penalization in the fitness function.
- With multiple objectives we find a pareto (set of non-dominated solutions).

HOW TO CONSTRUCT A GA?



Chromosomes selection

We must guarantee that the best individuals have a major possibility for being parents.

But, worse chromosomes must have an opportunity for reproduction. They can include useful genetic information in the reproduction process.

This idea define the “selective pressure”, that determines the degree of influence of the best individuals.

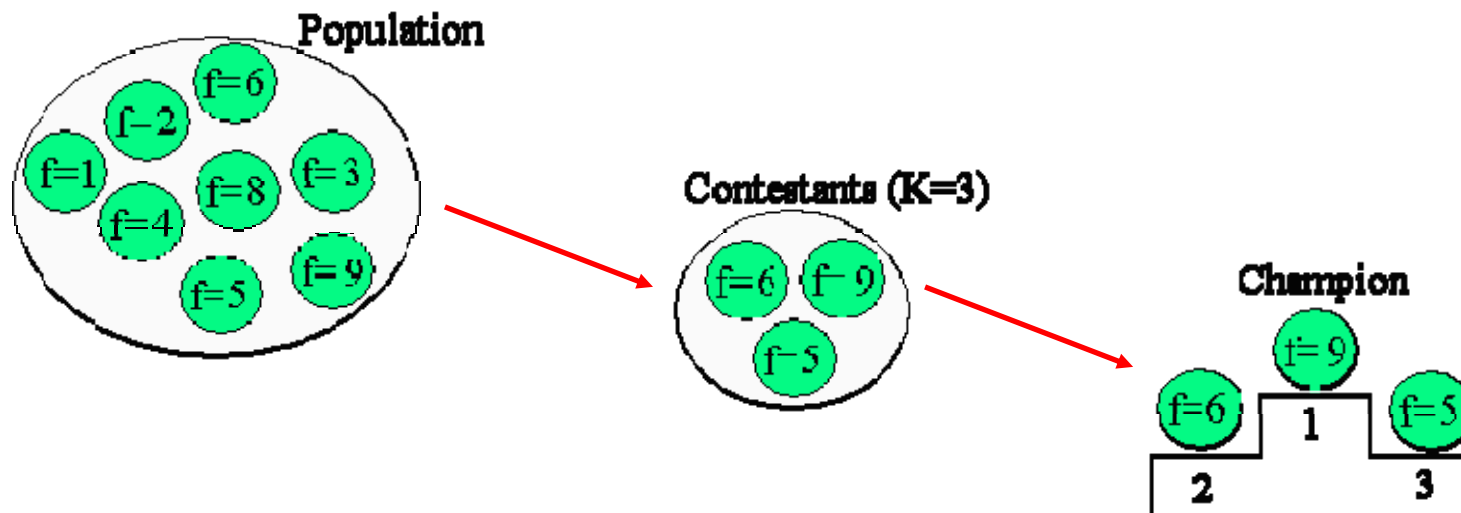
Strategy of selection

Tournament selection

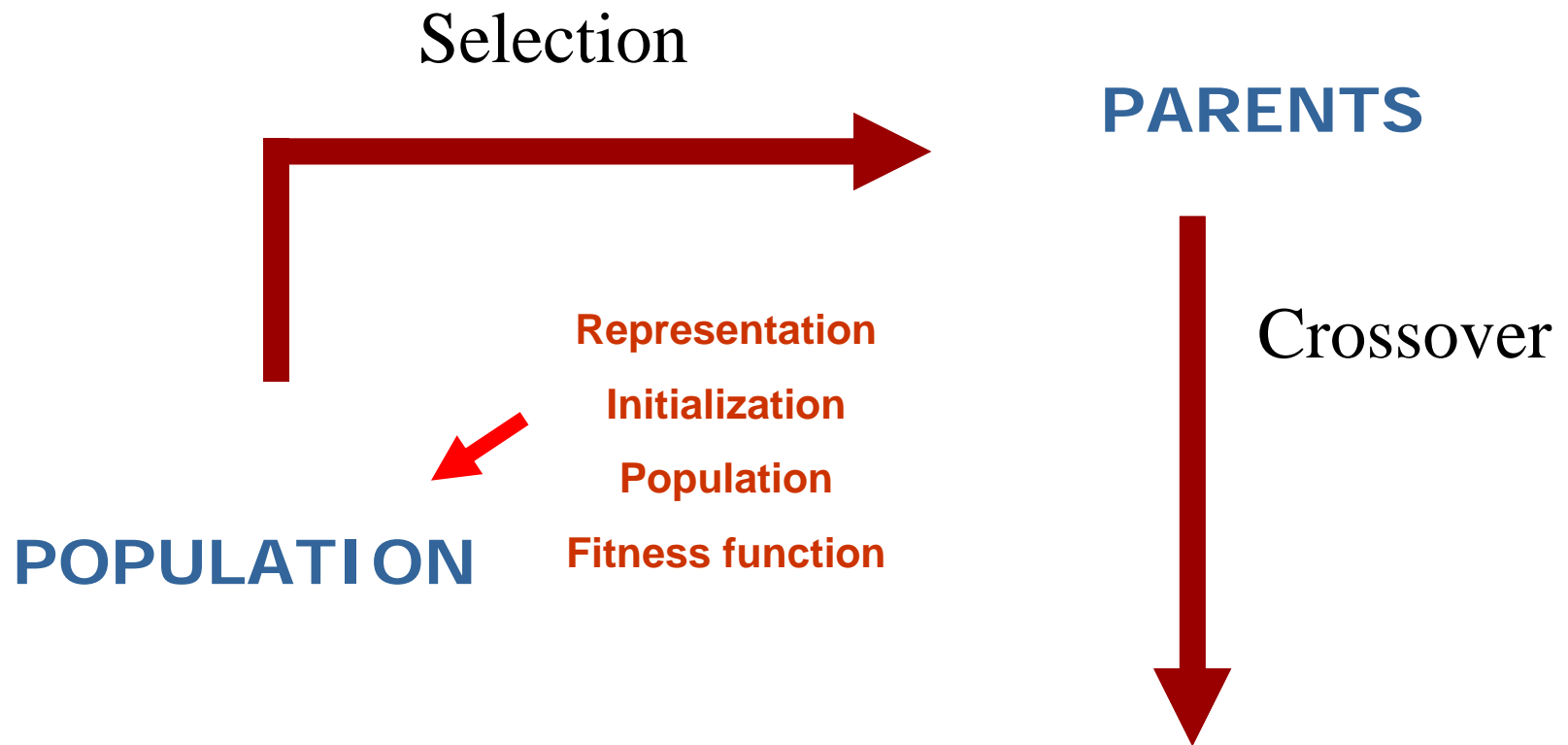
For each parent:

- Random selection of k individuals, with replacement
- Selection of the best

k is called the **tournament size**. A high k value, a high selective pressure and vice versa.



HOW TO CONSTRUCT A GA?



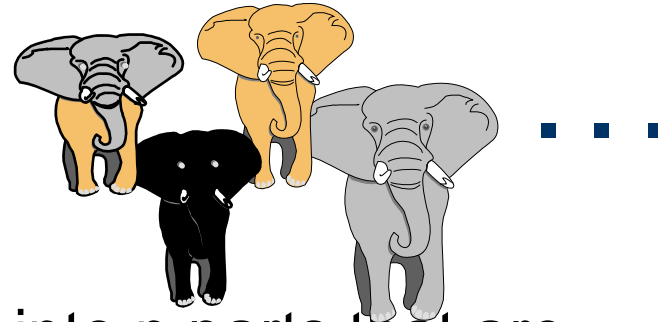
Crossover operator

Features:

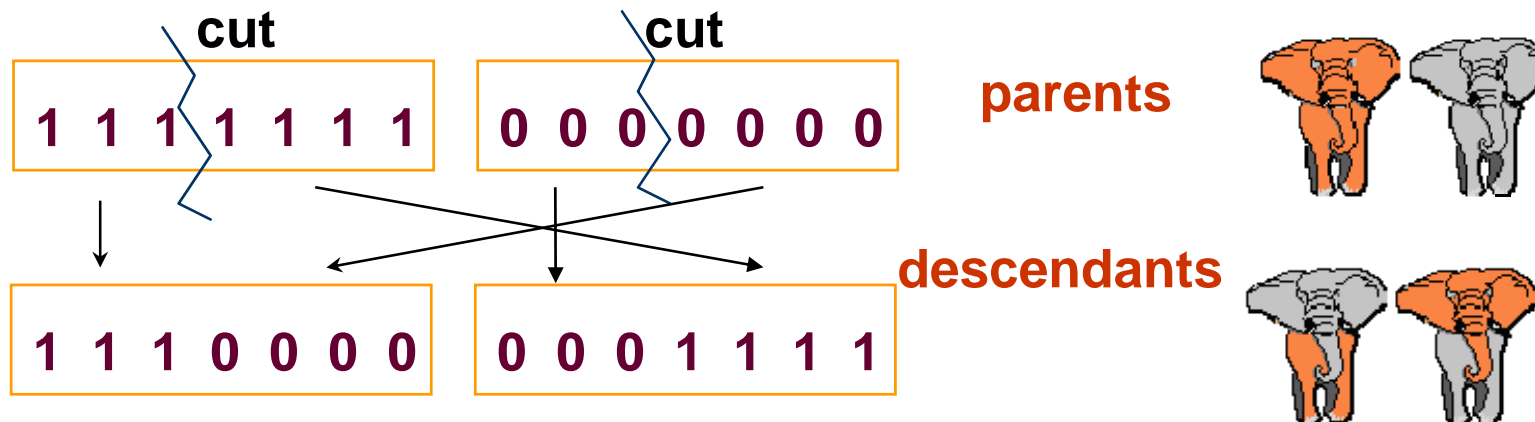
- The offspring must contain a heredity from the parents, associated to the parent features. In other case it would be a mutation operator.
- It depend on the representation.
- The recombination must produce valid chromosomes.
- It uses a probability for running on the two parents (P_c between 0.6 and 0.9, usually).

Example: Simple crossover on the binary representation

Population:

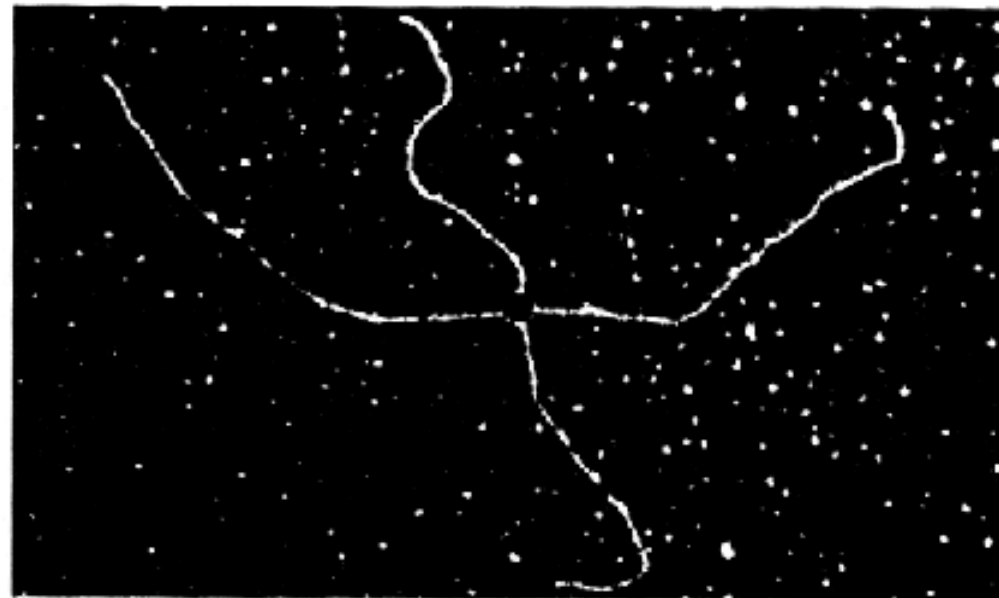
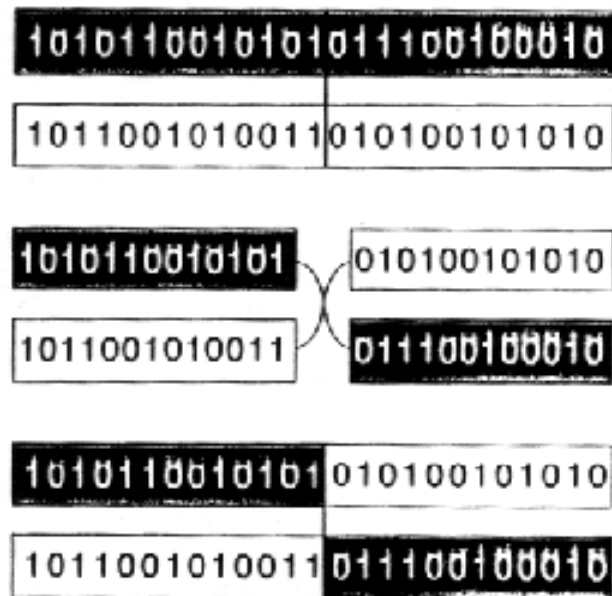


Each chromosome is divided into n parts that are recombined (example for $n = 2$)



Crossover operator

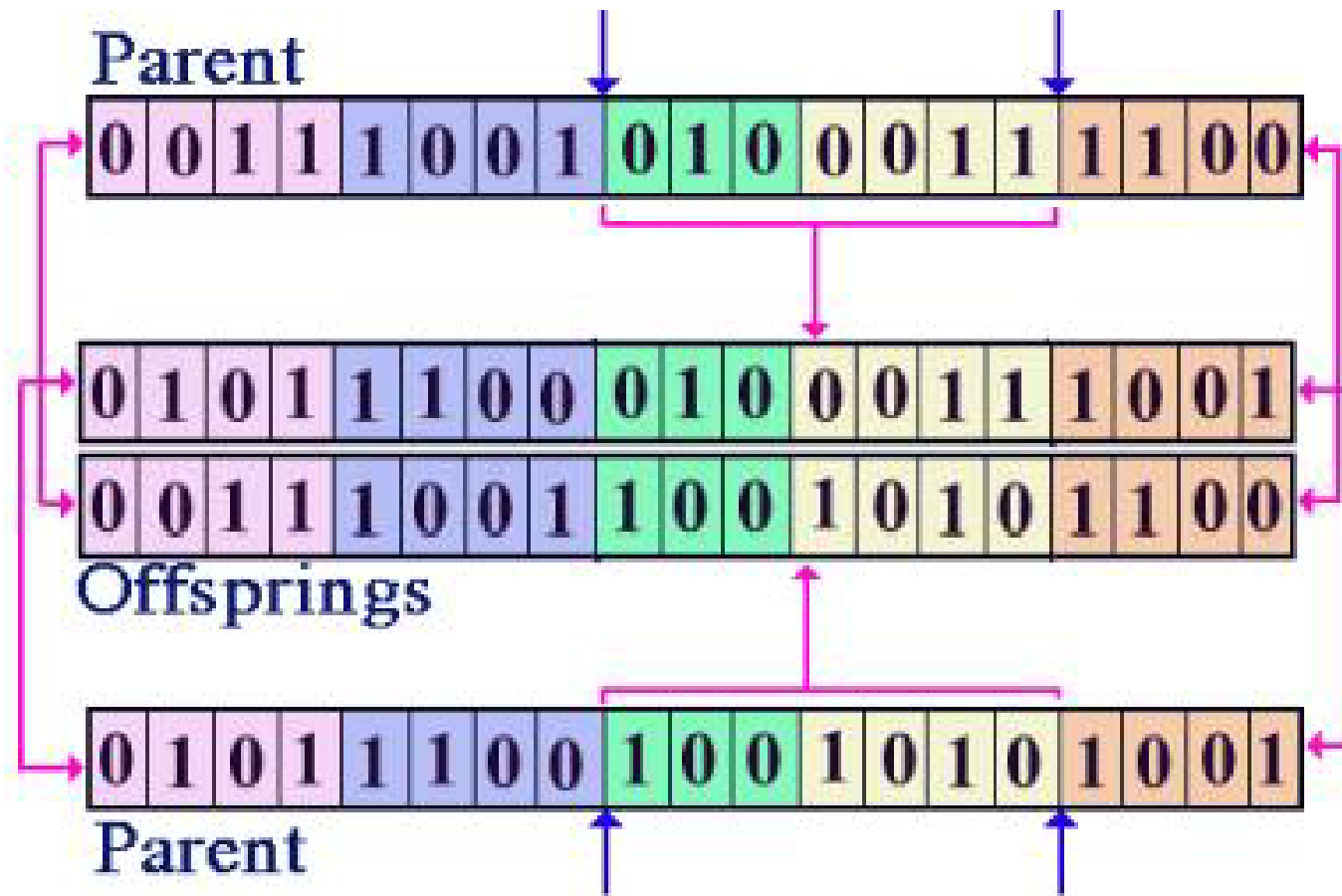
Classical image (John Holland): Biological crossover



CROSSOVER is the fundamental mechanism of genetic rearrangement for both real organisms and genetic algorithms.

Chromosomes line up and then swap the portions of their genetic code beyond the crossover point.

Example: Two points crossover



Example: uniform crossover

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

A	B	C	D	E	F	G	H
---	---	---	---	---	---	---	---



a	b	C	d	E	f	g	H
---	---	---	---	---	---	---	---

Example: Real coding crossover operator

Arithmetic crossover:

a	b	c	d	e	f
---	---	---	---	---	---

A	B	C	D	E	F
---	---	---	---	---	---



$(a+A)/2$	$(b+B)/2$	$(c+C)/2$	$(d+D)/2$	$(e+E)/2$	$(f+F)/2$
-----------	-----------	-----------	-----------	-----------	-----------

Example: real coding crossover operator **BLX- α**

- Two chromosomes

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

- BLX- α generates two descendants

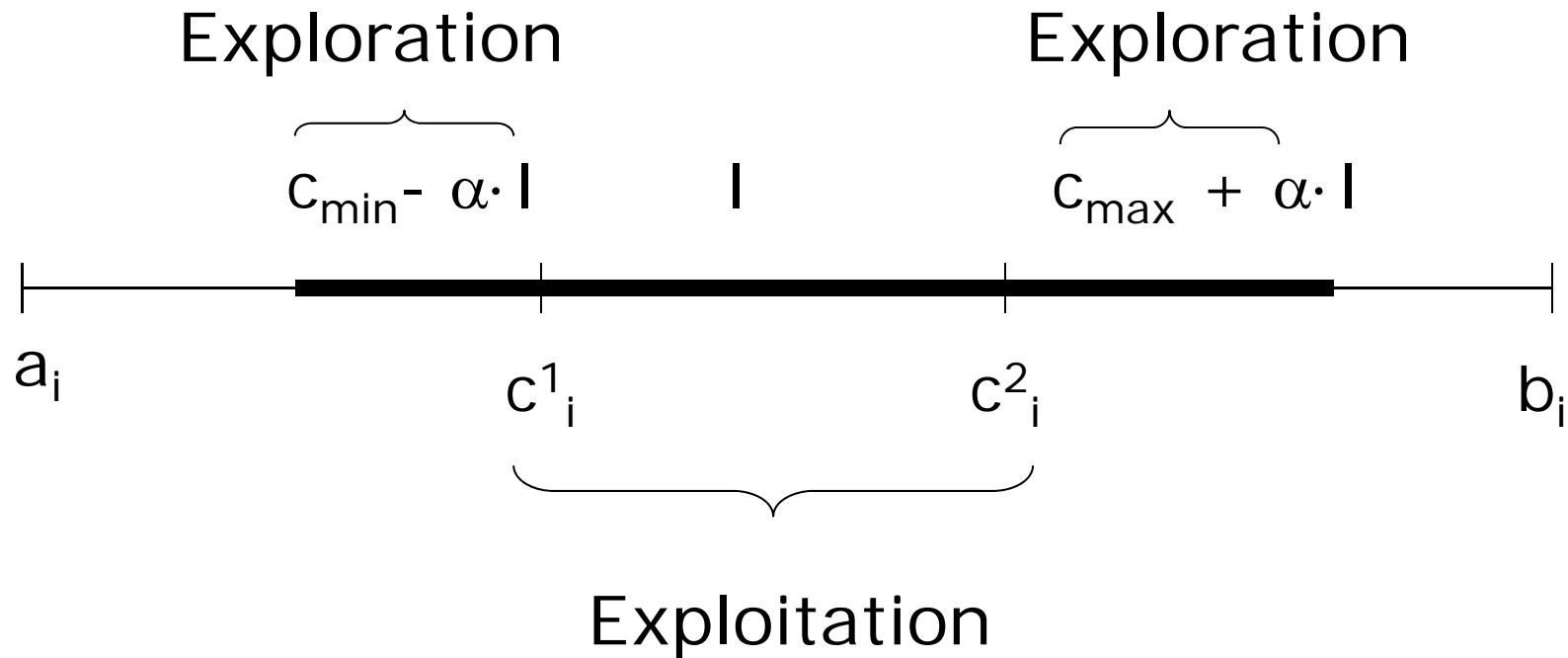
$$H_k = (h_{k1}, \dots, h_{ki}, \dots, h_{kn}) , k = 1, 2$$

- where h_{ki} is a random value in the interval:

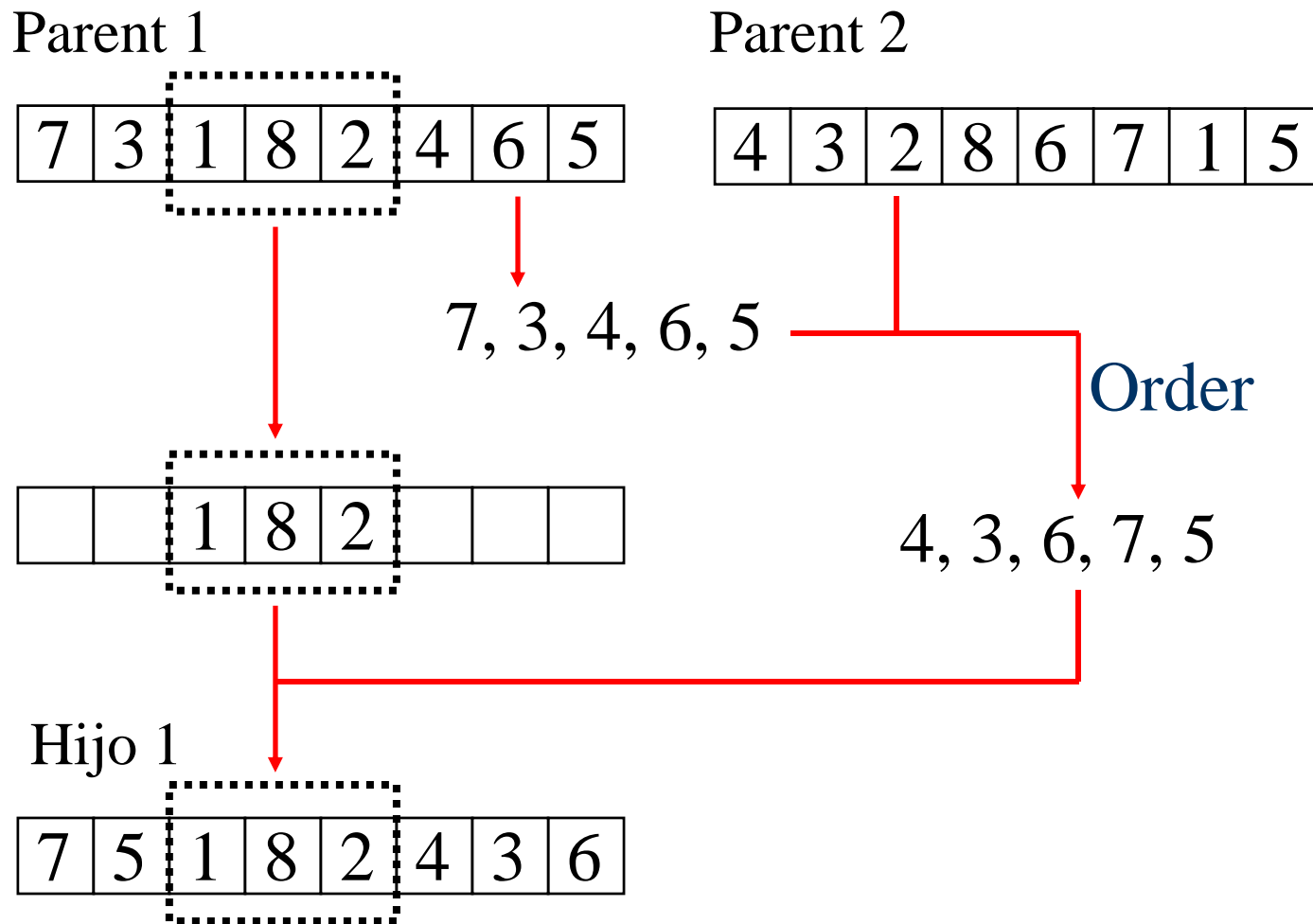
$$[C_{\min} - I \cdot \alpha, C_{\max} + I \cdot \alpha]$$

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

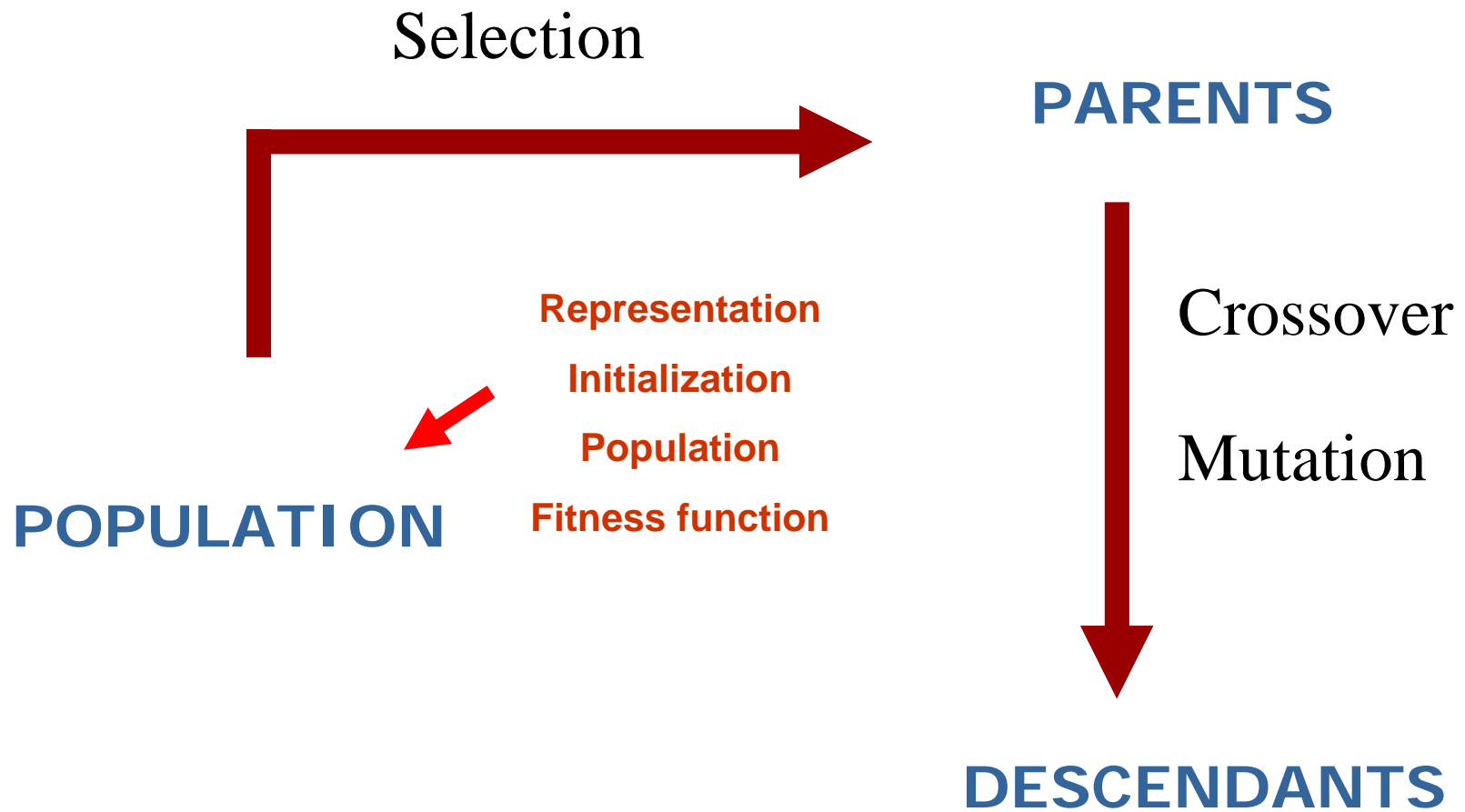
Ejemplo: Operador de cruce para representación real: **BLX- α**



Example: Crossover operator for order representation: OX



HOW TO CONSTRUCT A GA?

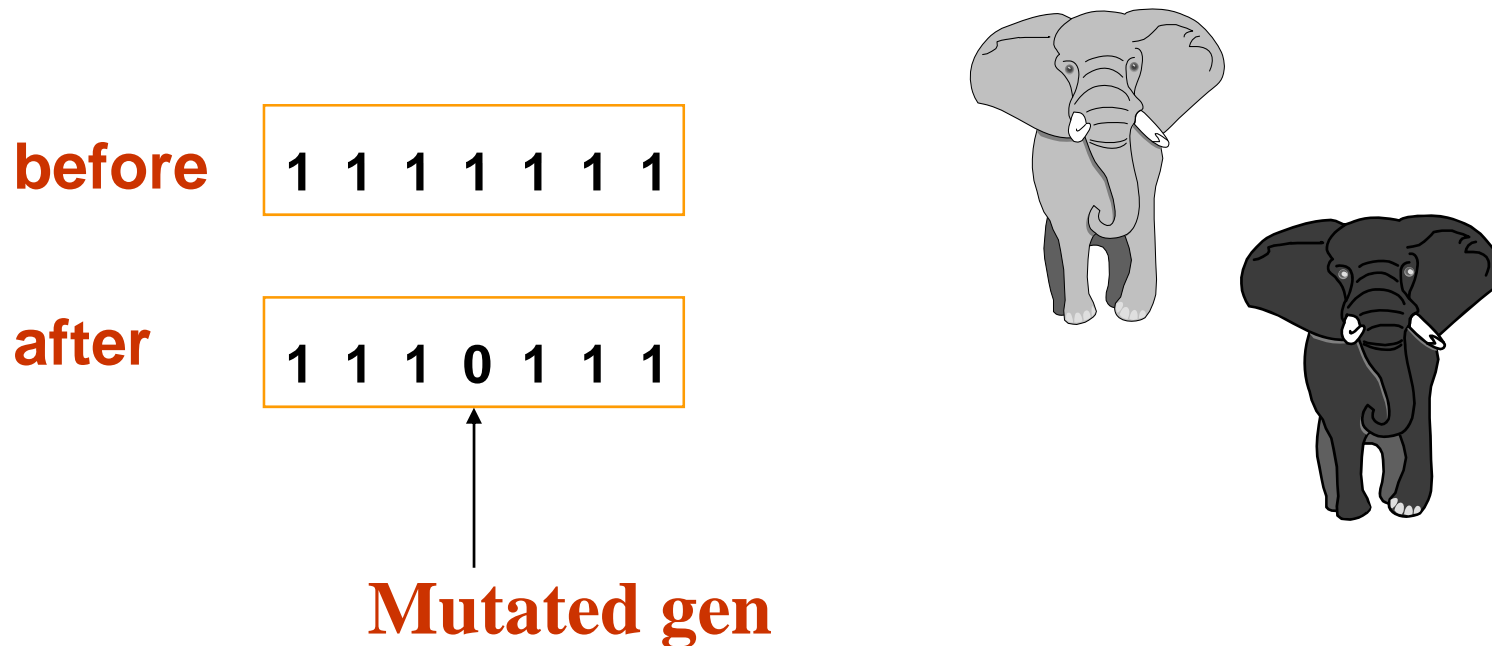


Mutation operator

Features:

- It must allow us to reach any point through a sequence of runs.
- We must control the size.
- It must produce valid chromosomes.
- It is used with a low running probability on the descendant obtained after the application of the crossover operator.

Example: binary mutation



The mutation happens with a low running probability per gen p_m

Example: real coding mutation

- Perturbation of real values via a random value.
- Using a gaussian/normal distribution $N(0, \sigma)$,
 - 0 is the mean
 - σ is the typical desviation

$$x'_i = x_i + N(0, \sigma_i)$$

For each parameter.

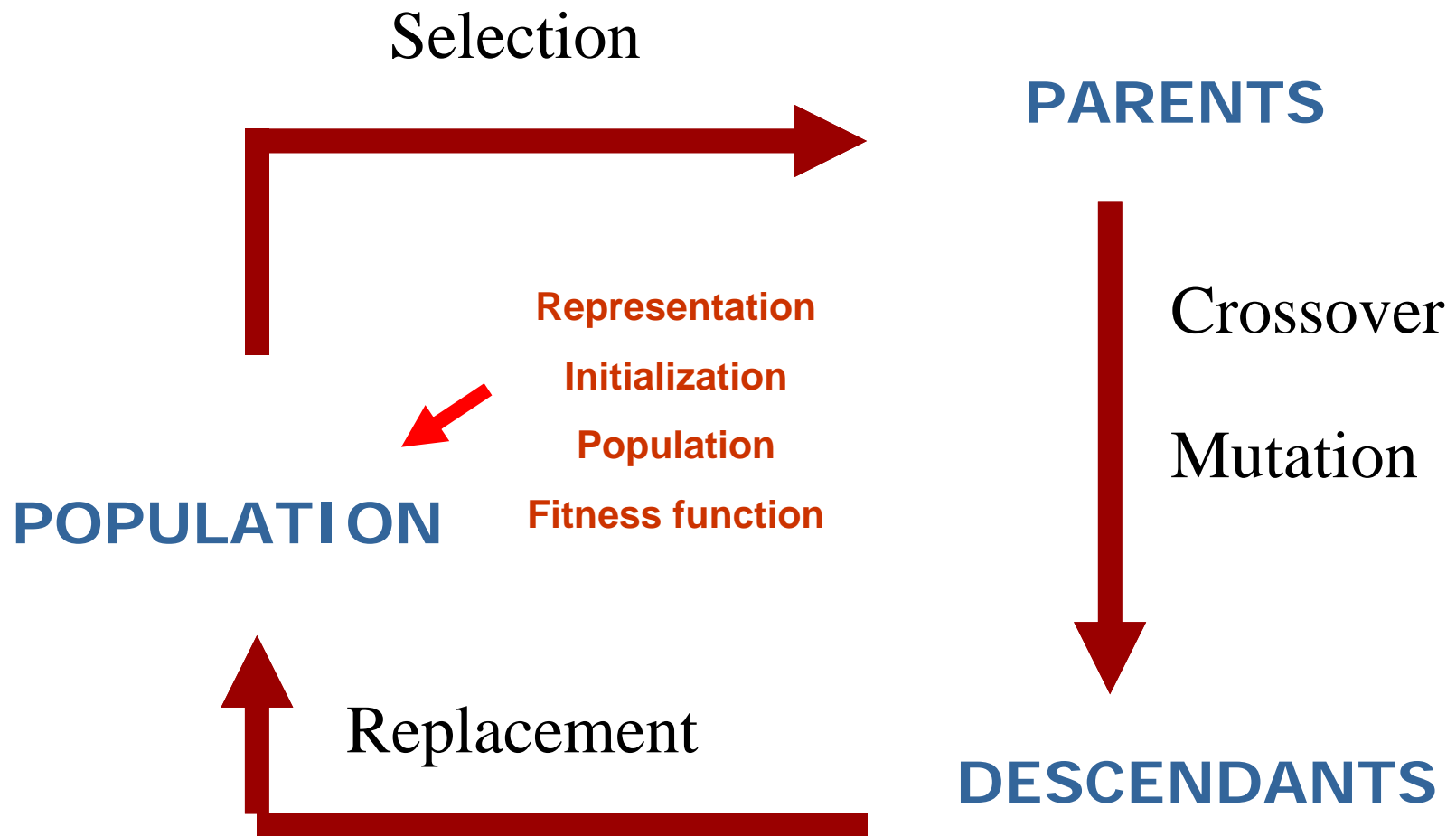
Example: order representation mutation

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



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

HOW TO CONSTRUCT A GA?



Replacement strategy

Complete population

Elitism: Maintaining the best chromosome

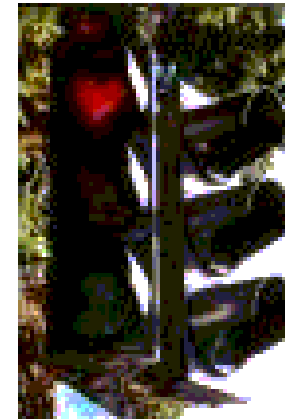
Replacement of parents per children (via competition)

....

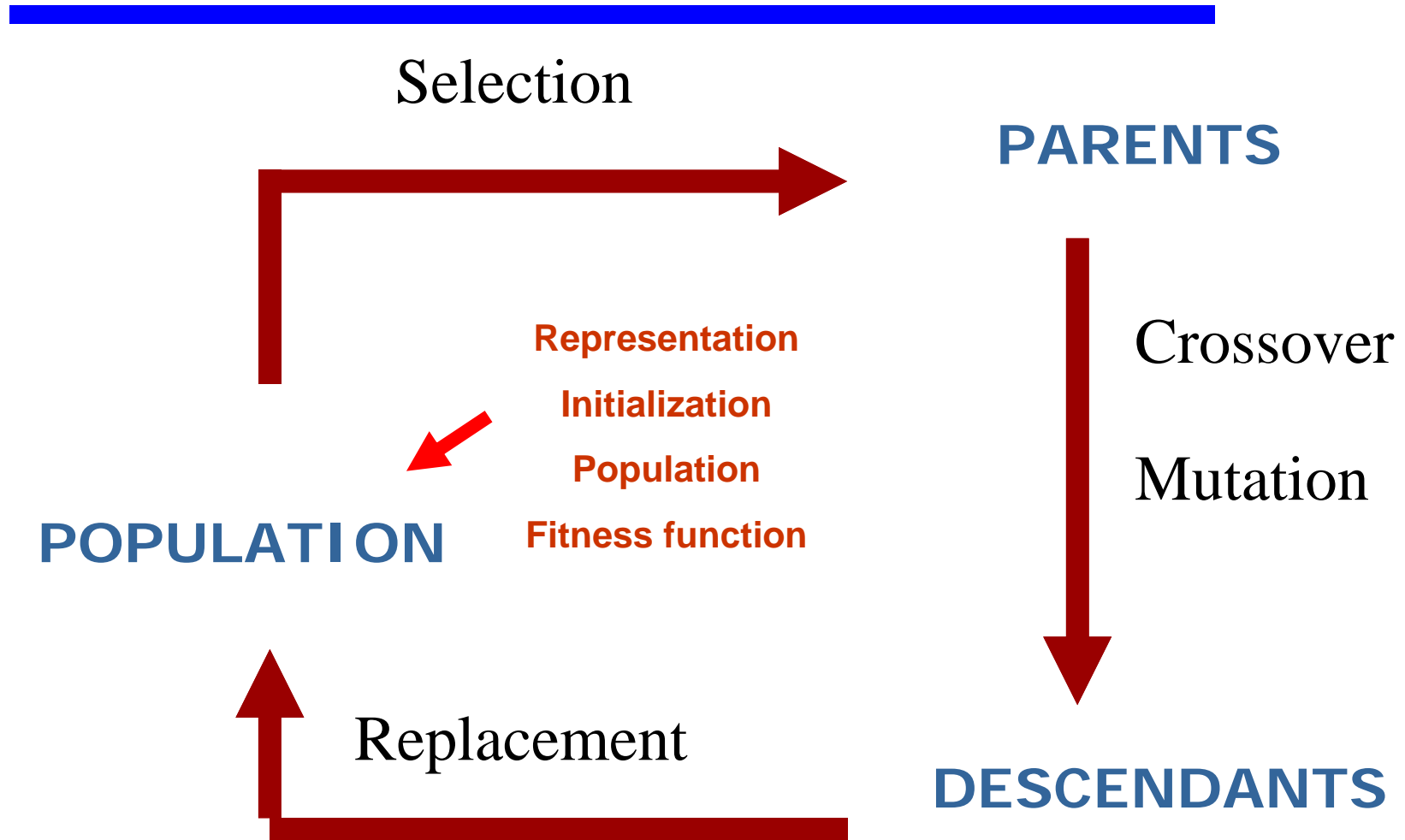
It depends on the model.

Stop condition

- ;When we reach the optimum!
- Limited CPU:
Maximum of evaluations
- After some iterations without any improvement.



Components



3. MODELS: GENERATIONAL vs STEADY STATE

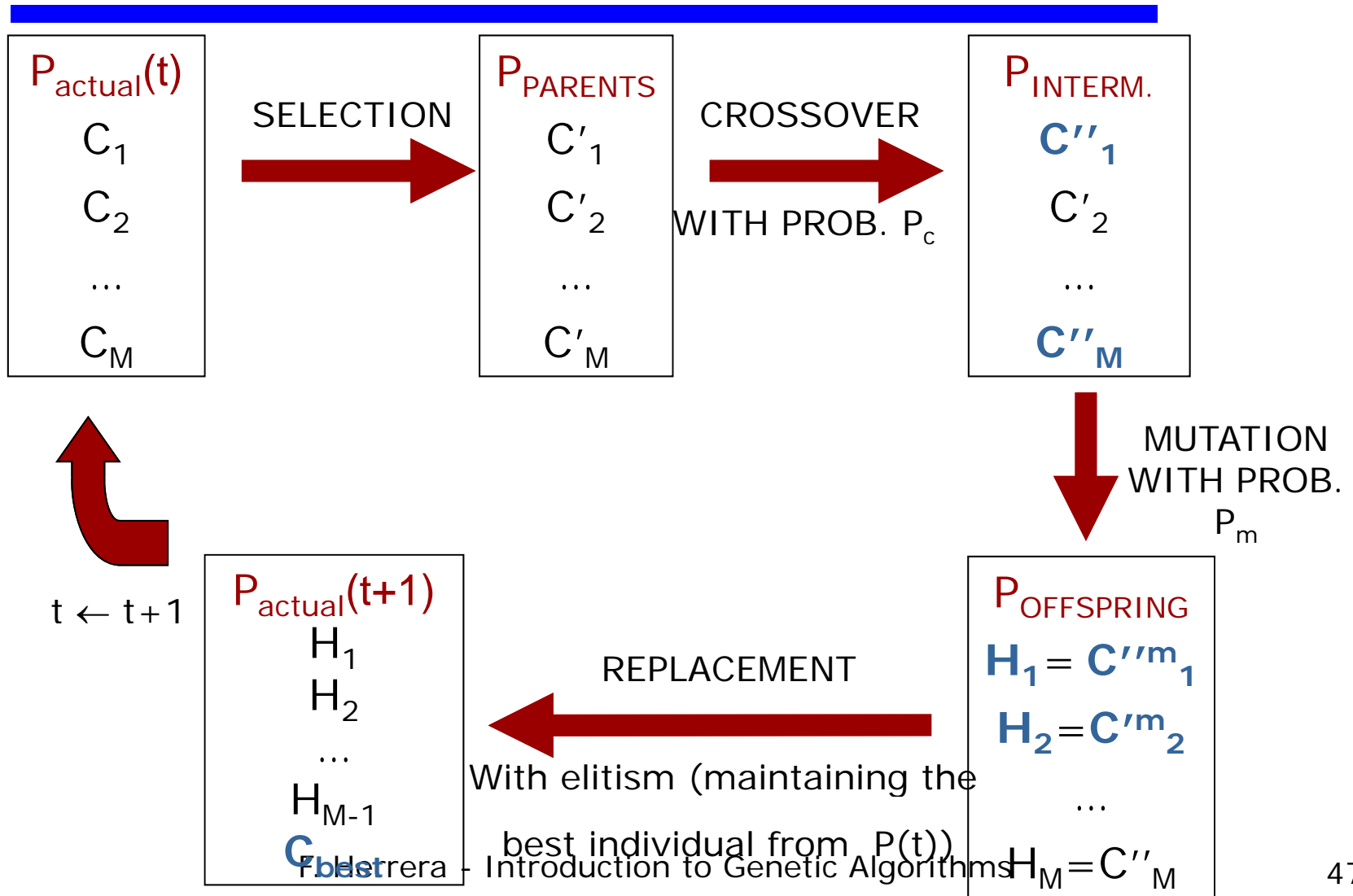
Generational model: Replacement of the complete population

Steady state model: Along each iteration two parents are used with genetic operators for getting one/two descendants.

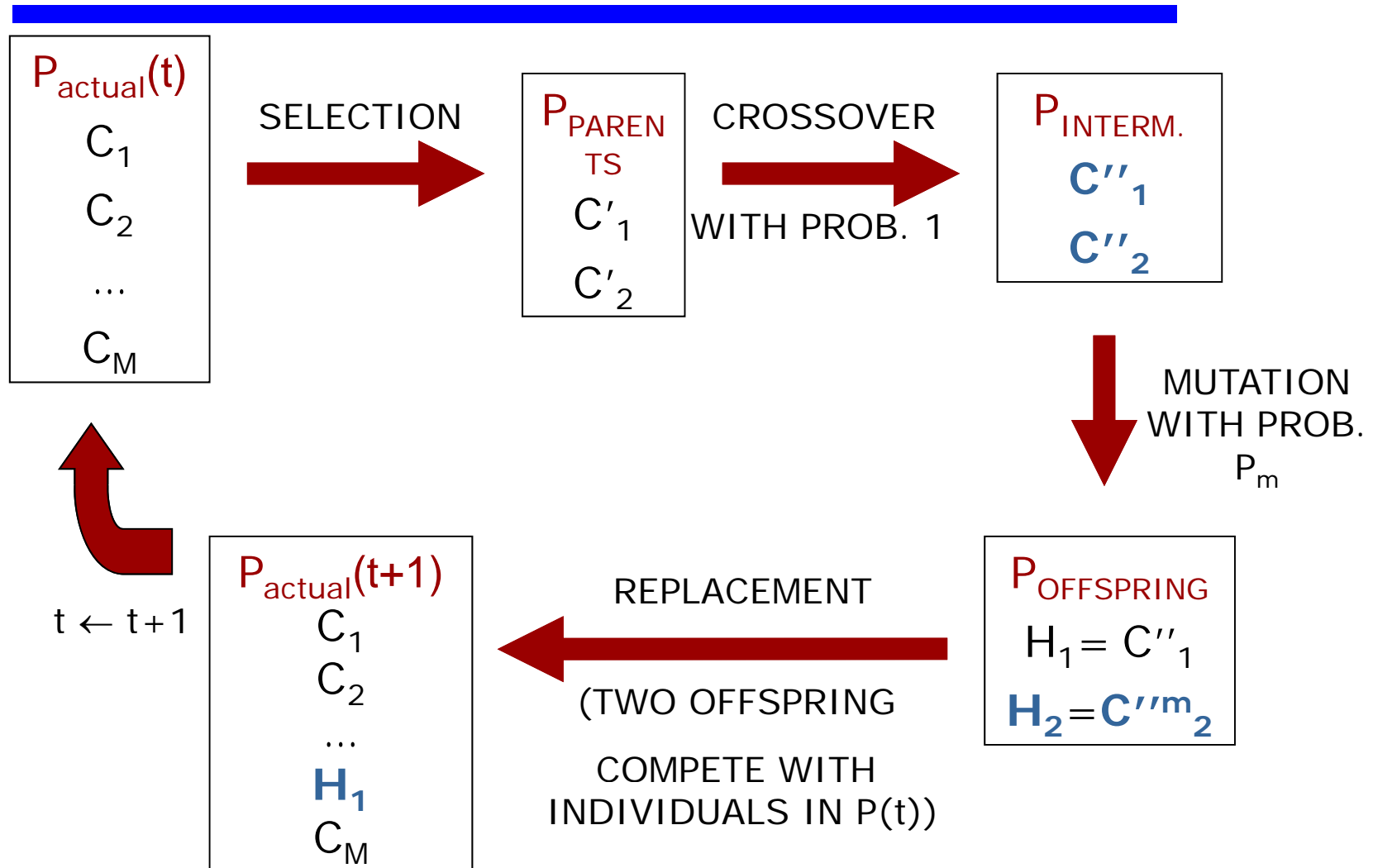
Only one/two individuals are replaced in the population.

The steady state is elitist. High selection pressure when we replace the worst individuals.

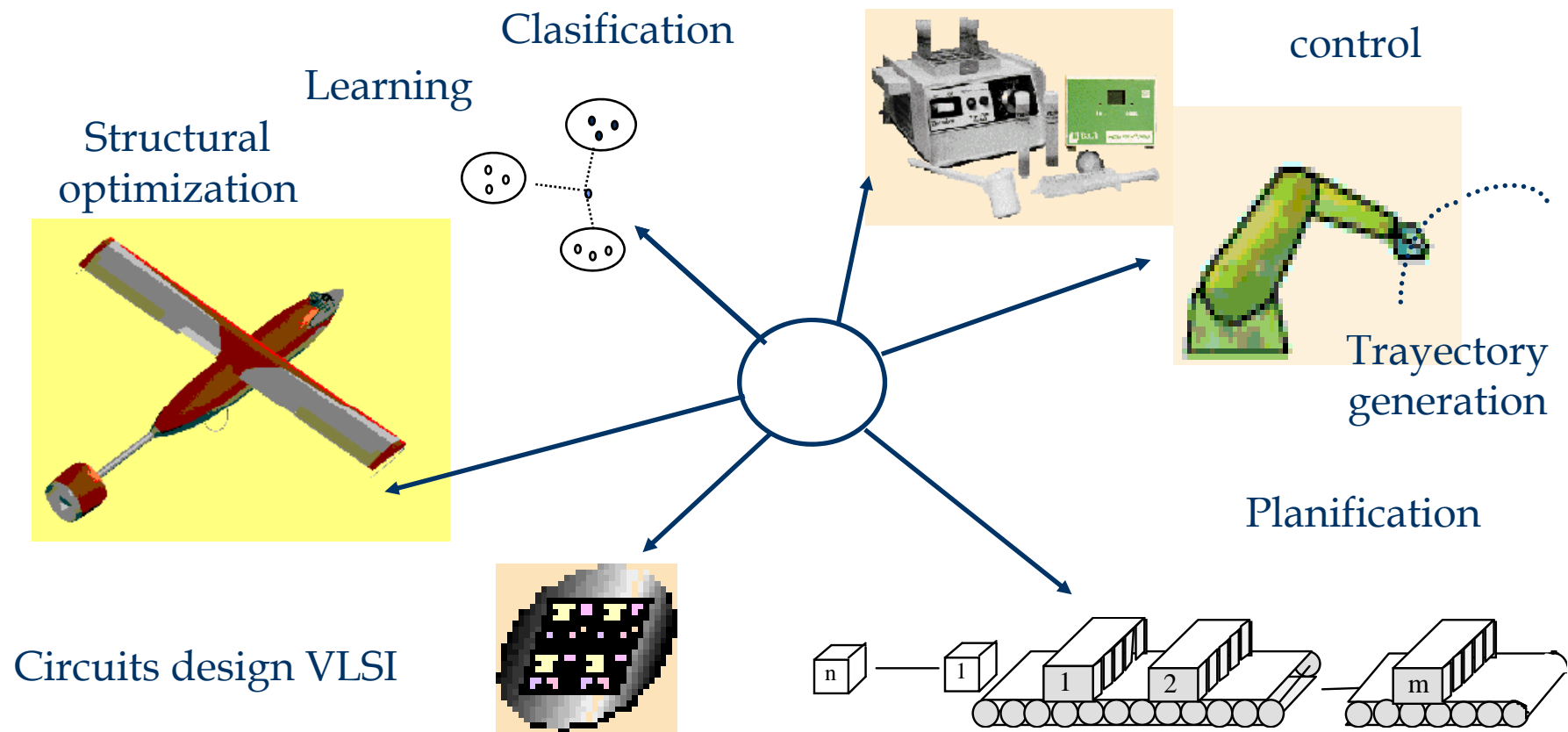
Generational model



Steady state model



4. APPLICATIONS



5. EXAMPLE: TRAVELLING SALESMAN PROBLEM

Order representation

(3 5 1 13 6 15 8 2 17 11 14 4 7 9 10 12 16)

17 cities

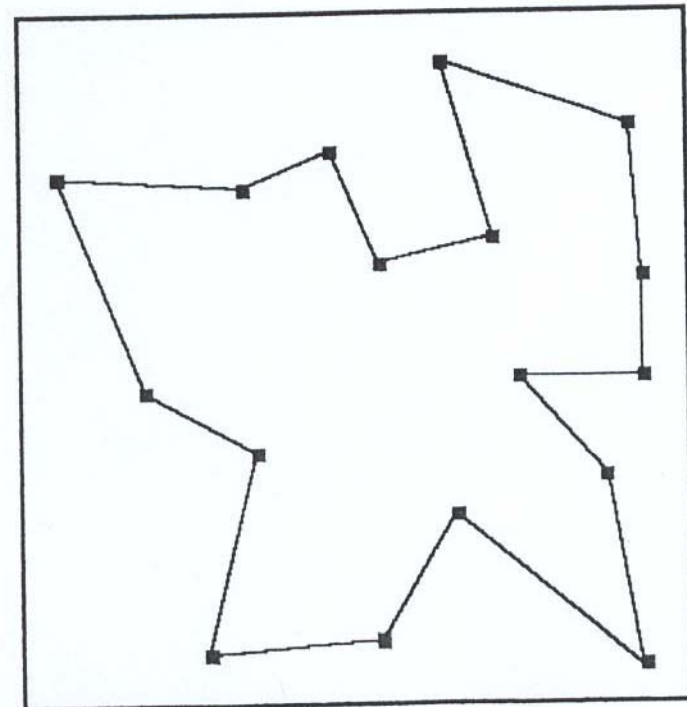
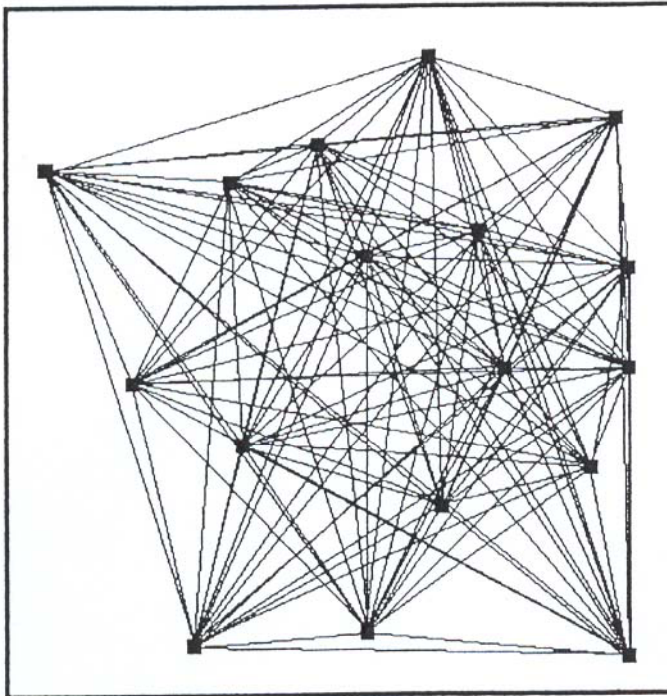
Objective: Sum of distance among cities.

Population: 61 chromosomes - Elitism

Crossover: OX ($P_c = 0,6$)

Mutation: List inversion ($P_m = 0,01$ – chromosome)

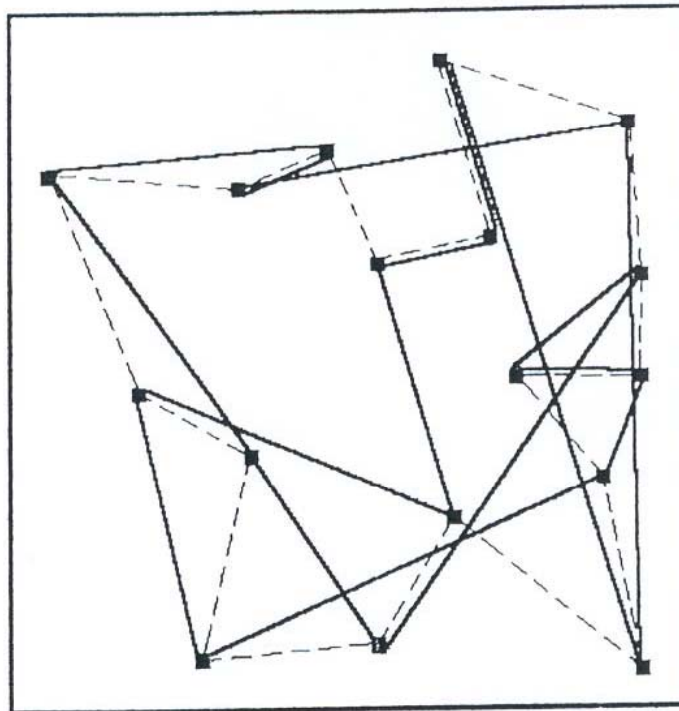
TSP



$17! = 3.5568743 \text{ e}14$ possible solutions

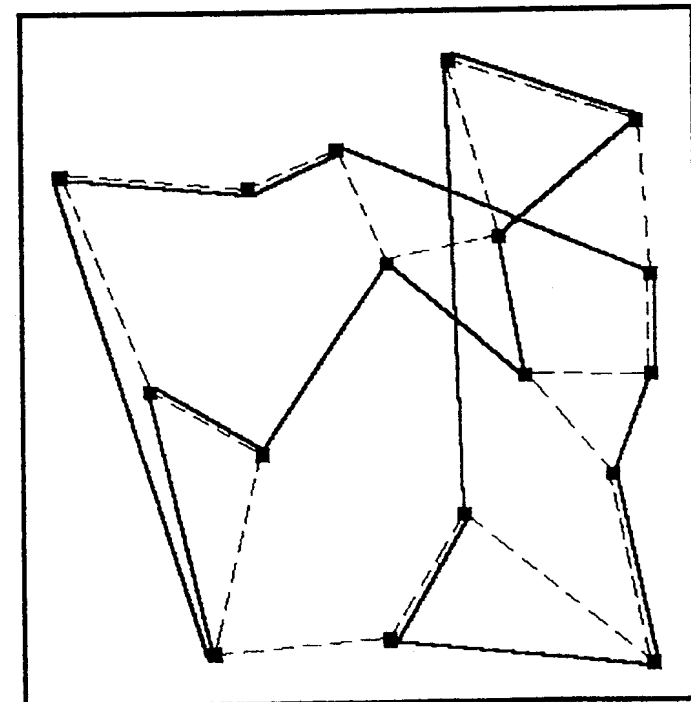
Optimum solution: 226.64

TSP



—— Mejor solución
----- Solución optimal

Iteration: 0 Cost: 403.7

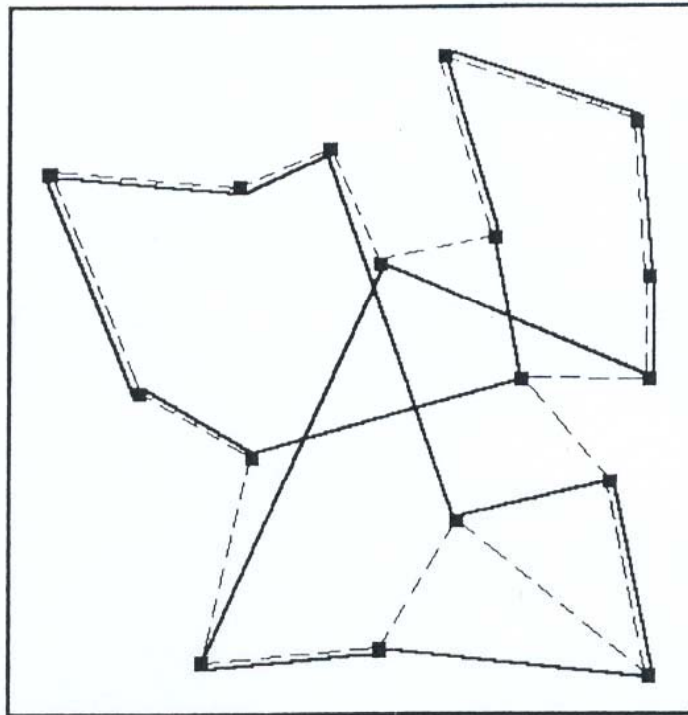


—— Mejor solución
----- Solución optimal

Iteration: 25 Cost: 303.86

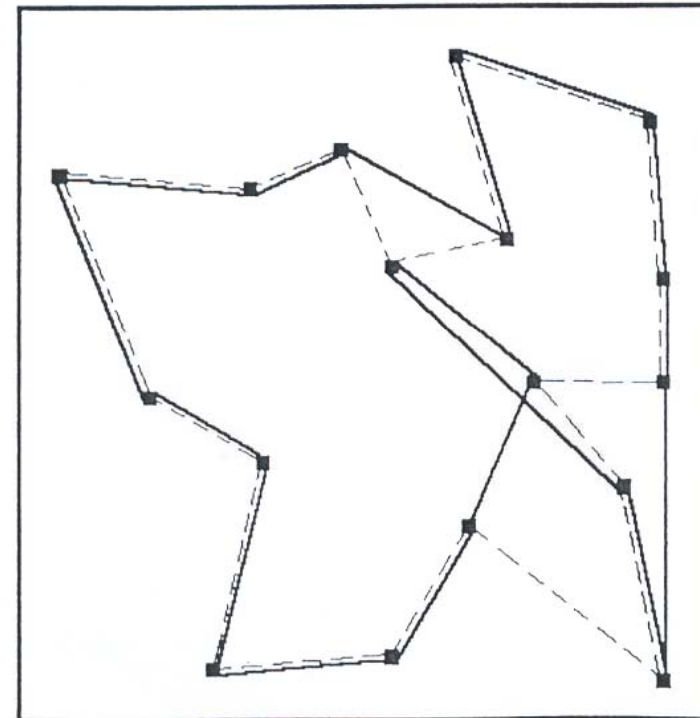
Optimum solution: 226.64

TSP



—— Mejor solución
- - - - Solución optimal

Iteration: 50 Cost: 293,6

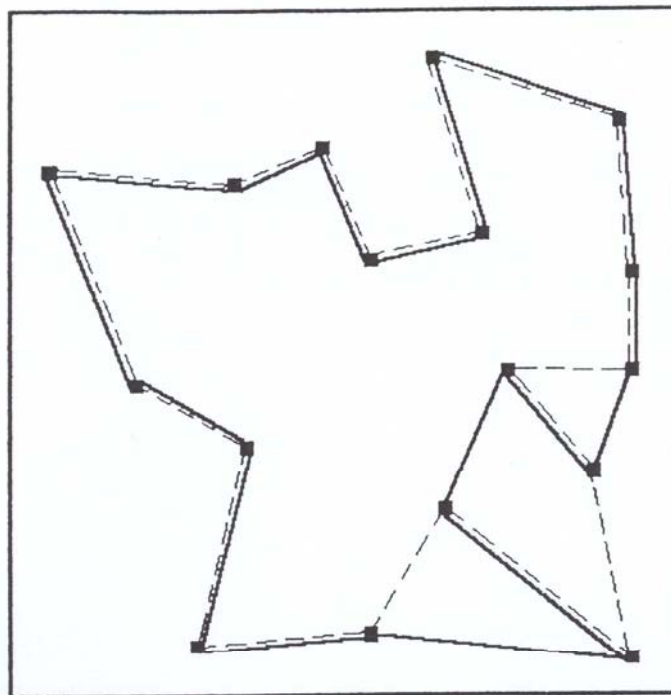


—— Mejor solución
- - - - Solución optimal

Iteration: 100 Cost: 256,55

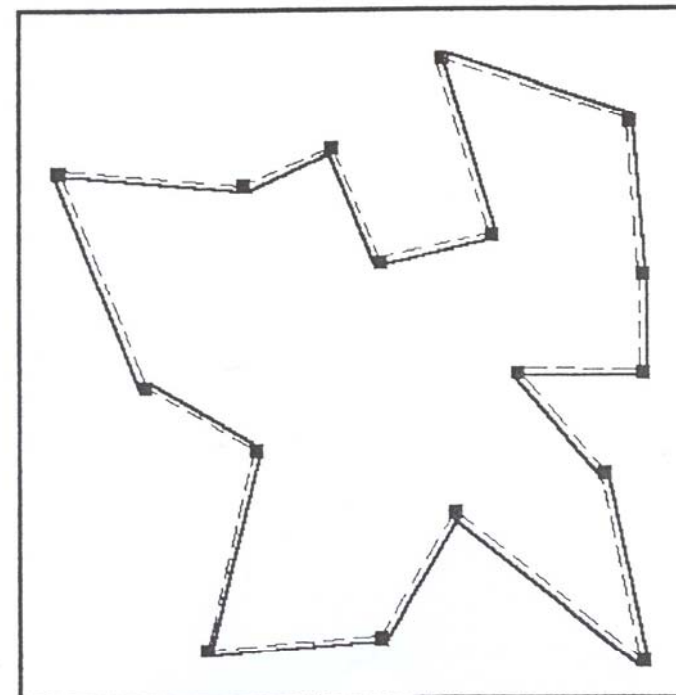
Optimum solution: 226,64

TSP



—— Mejor solución
- - - Solución optimal

Iteration: 200 Costo: 231,4

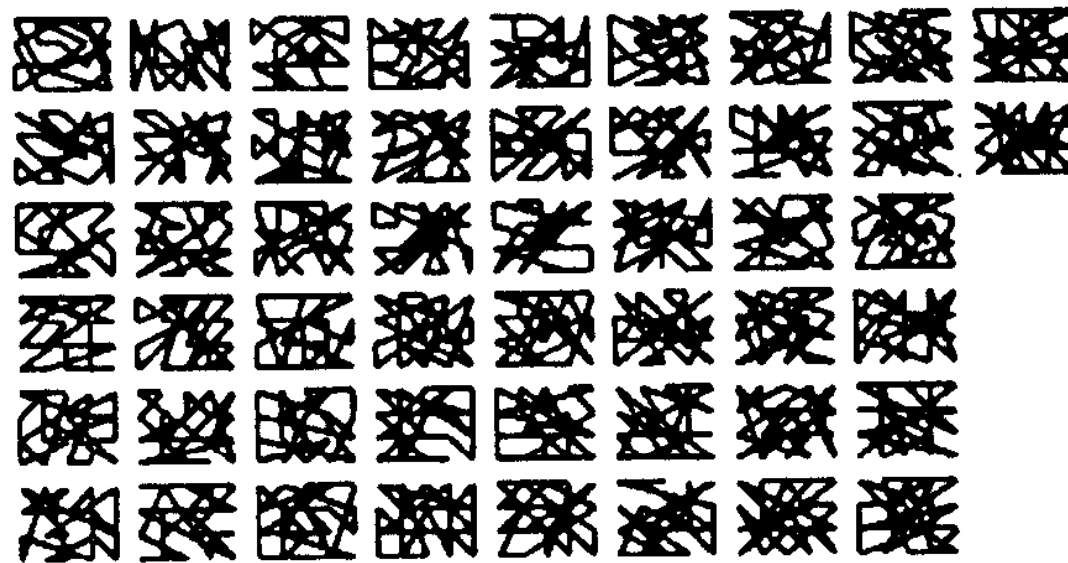


—— Mejor solución
- - - Solución optimal

Iteration: 250

Optimum solution: 226,64

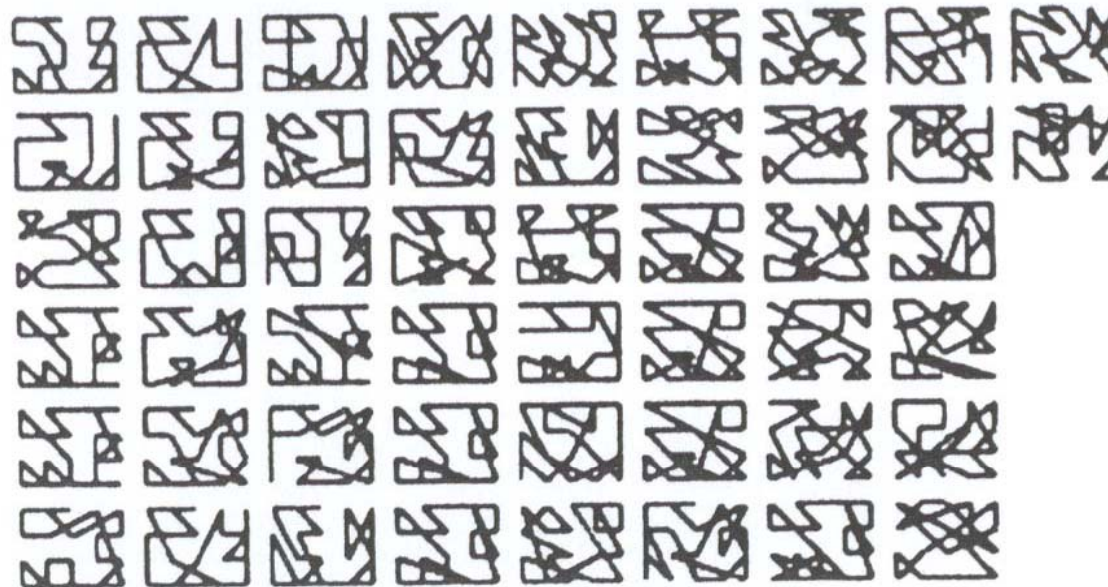
TSP



(0)

Visualization of the evolution with a population of size 50 and 70 iterations

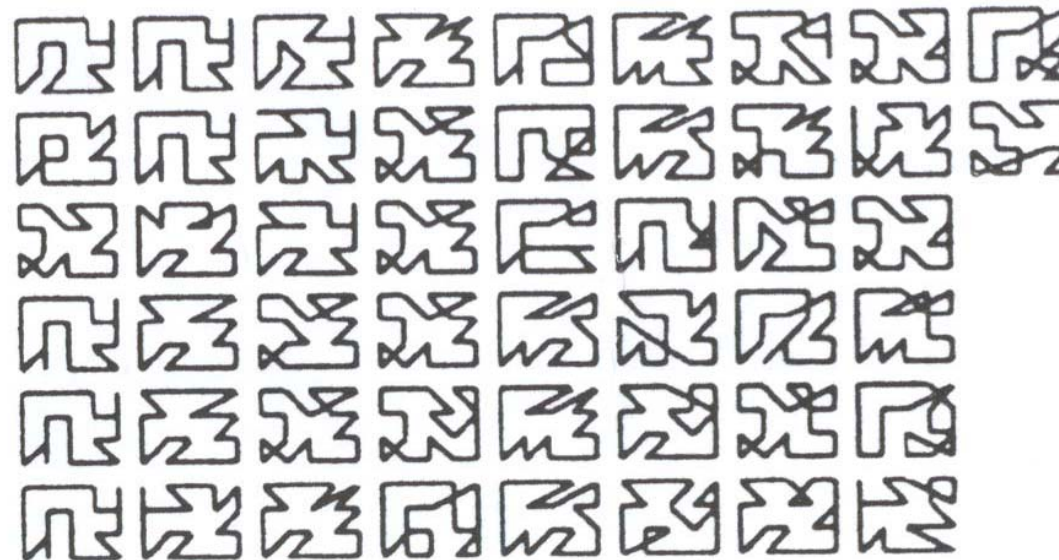
TSP



(10)

Visualization of the evolution with a population of size 50 and 70 iterations

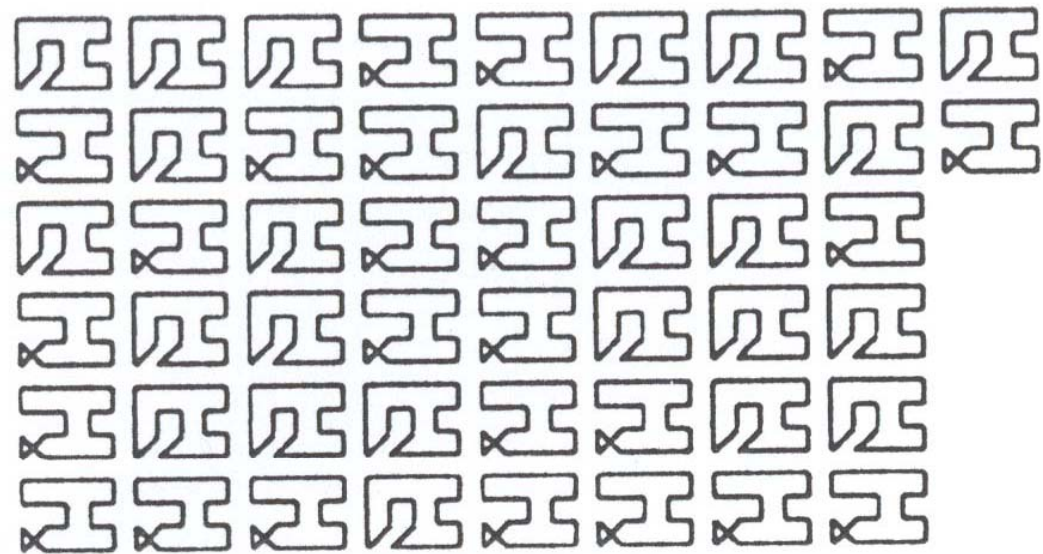
TSP



(30)

Visualization of the evolution with a population of size 50 and 70 iterations

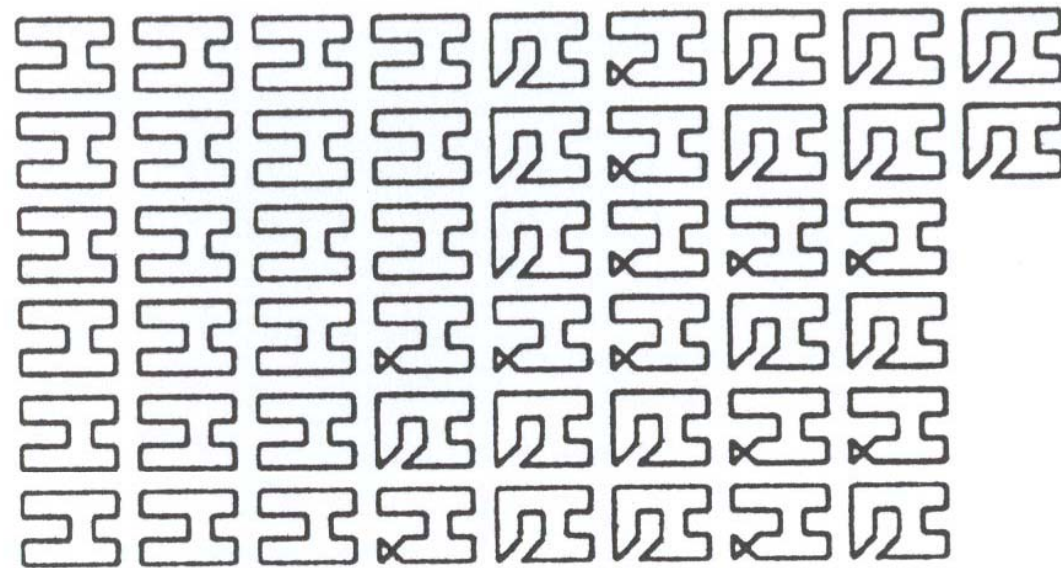
TSP



(50)

Visualization of the evolution with a population of size 50 and 70 iterations

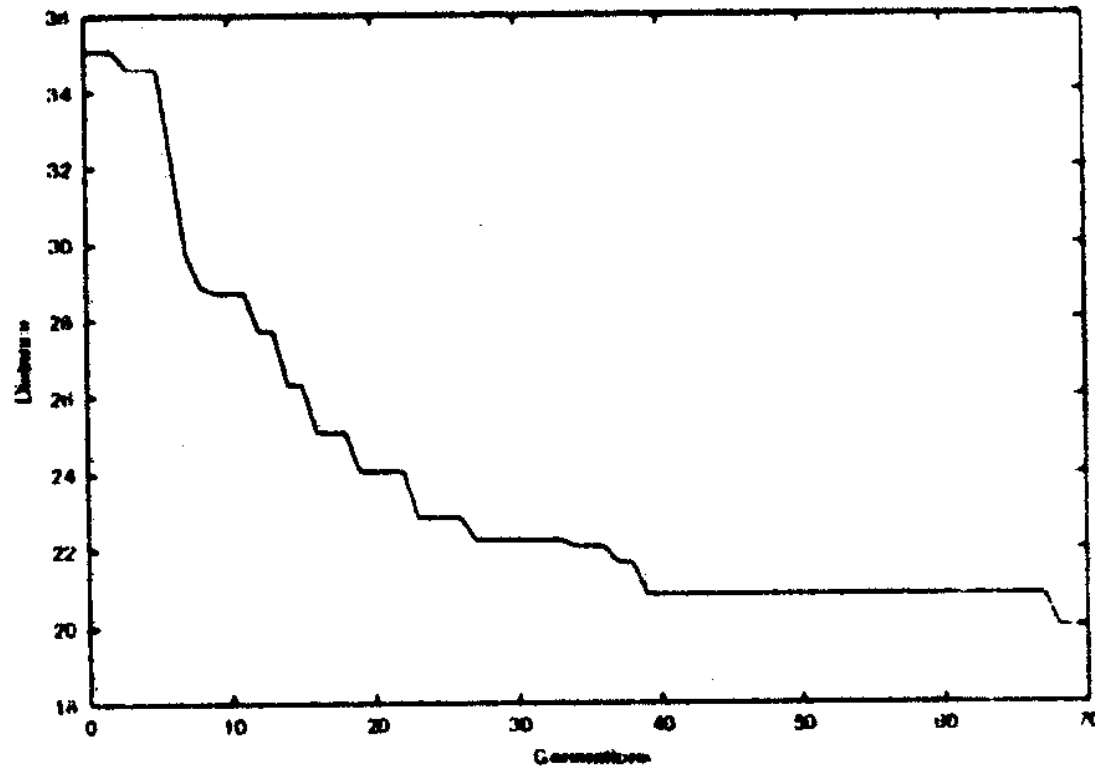
TSP



(70)

Visualization of the evolution with a population of size 50 and 70 iterations

TSP



Visualization of the evolution with a population of size 50 and 70 iterations

6. SOFTWARE AND IMPLEMENTATIONS

EO Evolutionary Computation Framework

EO is a template-based, ANSI-C++ compliant evolutionary computation library. It contains classes for almost any kind of evolutionary computation you might come up to at least for the ones we could think of. It is component-based, so that if you don't find the class you need in it, it is very easy to subclass existing abstract or concrete classes.

<http://eodev.sourceforge.net/>

Maintained by J.J. Merelo, Grupo Geneura, Univ. Granada
<jjmerelo@gmail.com>

6. SOFTWARE AND IMPLEMENTATIONS

JCLEC JAVA Library



JCLEC is a software system for Evolutionary Computation (EC) research, developed in the Java programming language. It provides a high-level software environment to do any kind of Evolutionary Algorithm (EA), with support for genetic algorithms (binary, integer and real encoding), genetic programming (Koza style, strongly typed, and grammar based) and evolutionary programming.

<http://jclec.sourceforge.net/>

Maintained: Sebastián Ventura, Universidad de Córdoba (sventura@uco.es)

S. Ventura, C. Romero, A. Zafra, J.A. Delgado, C. Hervás-Martínez. JCLEC: A Java Framework for Evolutionary Computing. *Soft Computing* 12:4 (2008) 381-392

6. SOFTWARE AND IMPLEMENTATIONS

Real Coding EAs implementations

The parameter optimization problems (continuous optimization) receive a special attention. There are specific Eas devoted to this kind of problems, such us: evolutionary strategies, differential evolution,

The following website is devoted to this problem, containing software for managing them:

<http://sci2s.ugr.es/EAMHCO/>



Software

Classic Evolutionary Algorithms

- **Real Evolutionary Algorithm (Realea) Library**
This library is created to give a framework for real coding optimizations using optimization algorithms, like Evolutionary Algorithms (LS) and Local Search Methods. Its aim is to allow the researcher to focus only on the implementation of its algorithm. It includes the source code of test suites indicated in this page.
[Source Code \(C++\)](mailto:dmolina@decsai.ugr.es) by D. Molina dmolina@decsai.ugr.es
- **Real Coding CHC Adaptive Algorithm**
This source code implements the real coding CHC algorithm proposed in *Eshelman, L. and Caruana, A. and Schaffer, J. D. Real-Coded Genetic Algorithms and Interval-Schemata. Foundation of Genetic Algorithms, 2, pages 187-202, 1993.*
[Source Code \(C++\)](mailto:dmolina@decsai.ugr.es) by D. Molina dmolina@decsai.ugr.es
- **Original Differential Evolution (DE)**
This source code implements the classic Differential Evolution (DE) proposed by R. Storn and K. Price in *Storn, R. and Price, K. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, 11, pages 341-359, 1997.*
[Source Code \(C++\)](mailto:dmolina@decsai.ugr.es) by D. Molina dmolina@decsai.ugr.es. It requires the [Realea Library](#)

CONCLUDING REMARKS

Genetic Algorithms

- *Based in a biological metaphor: evolution*
- *high applicability*
- *very popular*
- *High performance and low cost*
- *Powerful algorithms for a lot of applications*



CONCLUDING REMARKS

EVOLUTIONARY COMPUTATION

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC ALGORITHMS

GENETIC PROGRAMMING

EVOLUTION STRATEGIES

EVOLUTIONARY PROGRAMMING

OTHER EVOLUTIONARY MODELS

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS



BIBLIOGRAPHY

BOOKS - Classics

- D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.
- Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag, 1996.
- D.B. Fogel (Ed.) *Evolutionary Computation. The Fossil Record*.
(Selected Readings on the History of Evolutionary Computation).
IEEE Press, 1998.

BOOK – Recent

- A.E. Eiben, J.E. Smith. *Introduction to Evolutionary Computation*.
Springer Verlag 2003. (Natural Computing Series)

TUTORIALS: <http://sci2s.ugr.es/docencia/index.php> (link course)

D. Whitley "A Genetic Algorithm Tutorial". *Statistics and Computing*, 4 (1994) 65-85.

F. Herrera, M. Lozano, J.L. Verdegay, Tackling Real-Coded Genetic Algorithms:
Operators and tools for the Behaviour Analysis. *Artificial Intelligence Review* 12
(1998) 265-319.