UNIVERSITÀ DI PISA

Dottorato di Ricerca in Ingegneria dell'Informazione

Data Mining and Soft Computing

Francisco Herrera

Research Group on Soft Computing and Information Intelligent Systems (SCI²S)

Dept. of Computer Science and A.I.

University of Granada, Spain

Email: herrera@decsai.ugr.es

http://sci2s.ugr.es

http://decsai.ugr.es/~herrera

DECSAI

Universidad de Granada

Summary

1. Introduction to Data Mining and Knowledge Discovery
2. Data Preparation
3. Introduction to Prediction, Classification, Clustering and Association
4. Data Mining - From the Top 10 Algorithms to the New Challenges
5. Introduction to Soft Computing. Focusing our attention in Fuzzy Logic and Evolutionary Computation
6. Soft Computing Techniques in Data Mining: Fuzzy Data Mining and Knowledge Extraction based on Evolutionary Learning
7. Genetic Fuzzy Systems: State of the Art and New Trends
8. Some Advanced Topics I: Classification with Imbalanced Data Sets
9. Some Advanced Topics II: Subgroup Discovery
10. Some advanced Topics III: Data Complexity
11. Final talk: How must I Do my Experimental Study? Design of Experiments in Data Mining/Computational Intelligence. Using Non-parametric Tests. Some Cases of Study.

Introduction to Soft Computing. Focusing our attention in Fuzzy Logic and Evolutionary Computation

Outline

- ✓ Introduction to Soft Computing/Computational Intelligence
- ✓ Fuzzy Sets and Fuzzy Systems
- ✓ Genetic Algorithms
- ✓ Neural Netwoks
- ✓ Concluding Remarks

Introduction to Soft Computing. Focusing our attention in Fuzzy Logic and Evolutionary Computation

Outline

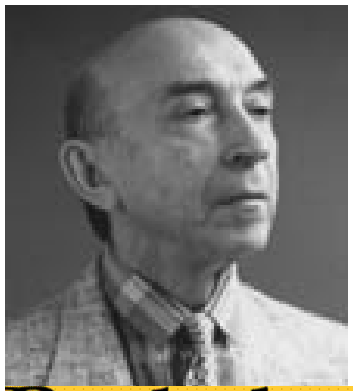
- ✓ Introduction to Soft Computing/Computational Intelligence
- ✓ Fuzzy Sets and Fuzzy Systems
- ✓ Genetic Algorithms
- ✓ Neural Netwoks
- ✓ Concluding Remarks

Soft Computing

Soft computing refers to a collection of computational techniques in computer science, machine learning and some engineering disciplines, which study, model, and analyze very complex phenomena: those for which more conventional methods have not yielded low cost, analytic, and complete solutions.

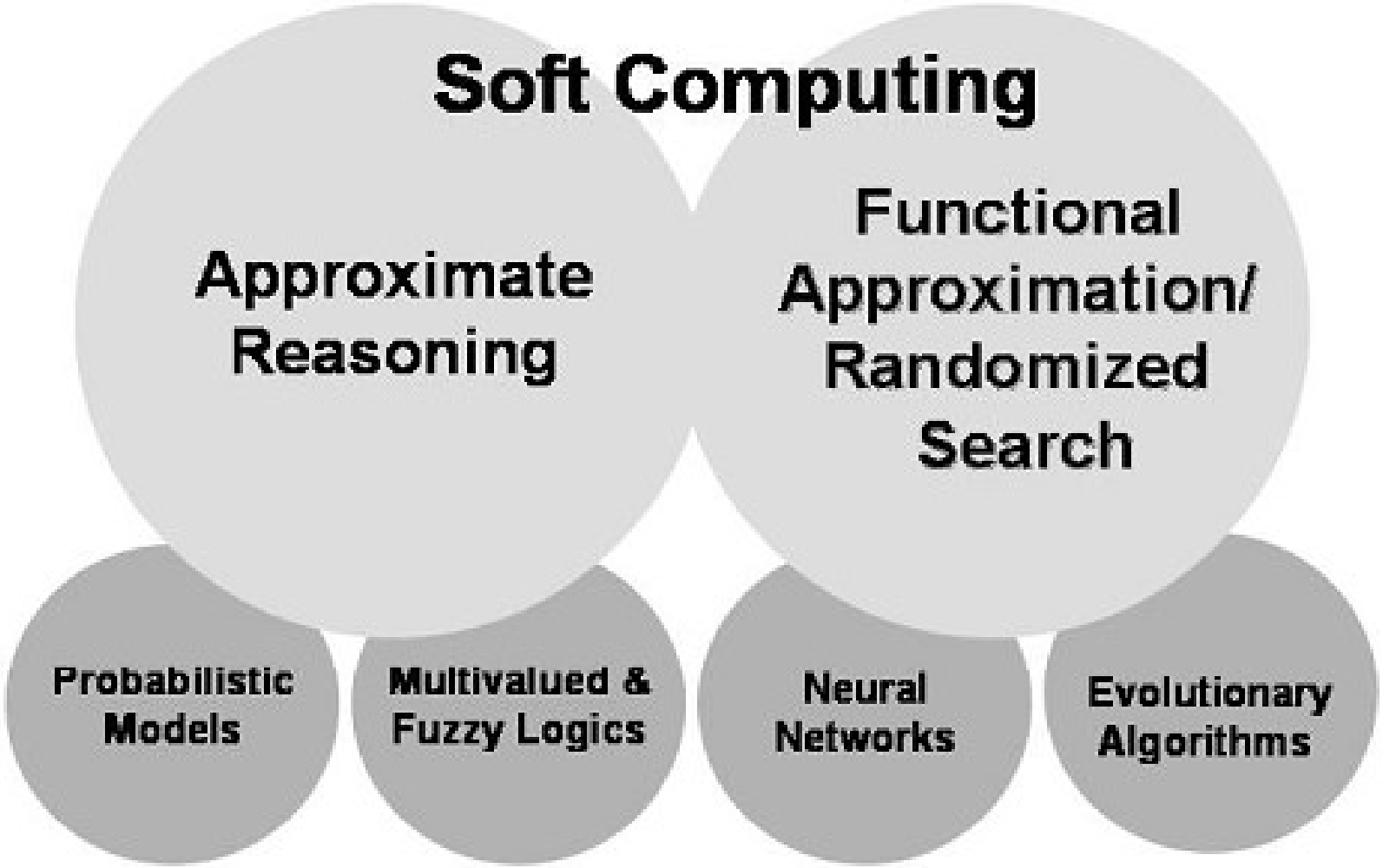
Prof. Zadeh:

"...in contrast to traditional hard computing, soft computing exploits the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, low solution-cost, and better rapport with reality"



Lotfi A. Zadeh
Introduce "Fuzzy Logic" in 1965 and "Soft Computing" in 1992.

Soft Computing



Computational Intelligence

The Field of Interest of the Society shall be the theory, design, application, and development of biologically and linguistically motivated computational paradigms emphasizing neural networks, connectionist systems, genetic algorithms, evolutionary programming, fuzzy systems, and hybrid intelligent systems in which these paradigms are contained.

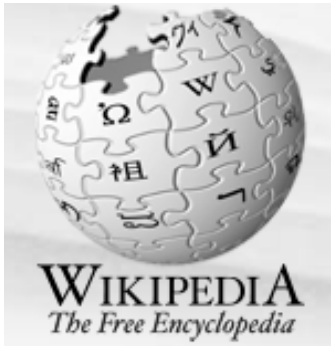
Computational Intelligence

Pedrycz's Definition of Computational Intelligence

Computational intelligence (CI) is a recently emerging area of fundamental and applied research exploiting a number of advanced information processing technologies. The main components of CI encompass neural networks, fuzzy set technology and evolutionary computation. In this triumvirate, each of them plays an important, well-defined, and unique role.

(Pedrycz 1998)

Computational Intelligence



Computational intelligence

From Wikipedia, the free encyclopedia

http://en.wikipedia.org

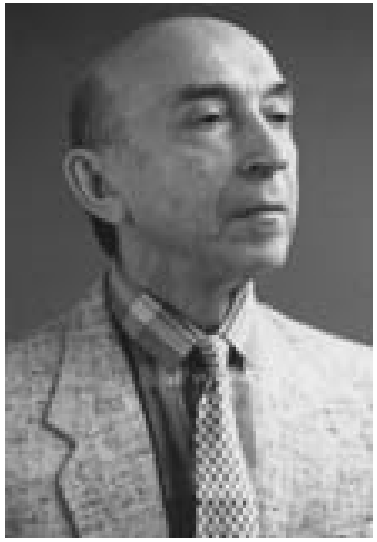
"Computational intelligence (CI) is a successor of artificial intelligence. As an alternative to GOFAI it rather relies on heuristic algorithms such as in fuzzy systems, neural networks and evolutionary computation. In addition, computational intelligence also embraces techniques that use Swarm intelligence, Fractals and Chaos Theory, Artificial immune systems, Wavelets, etc."

"GOFAI stands for "Good Old-Fashioned Artificial Intelligence". It is commonly used to denote a branch of Artificial Intelligence which mainly deals with symbolic problems. It is based on the assumption that thinking is nothing but symbol manipulation. Thus, it holds out the hope that computers will not merely simulate intelligence, but actually achieve it."

Outline

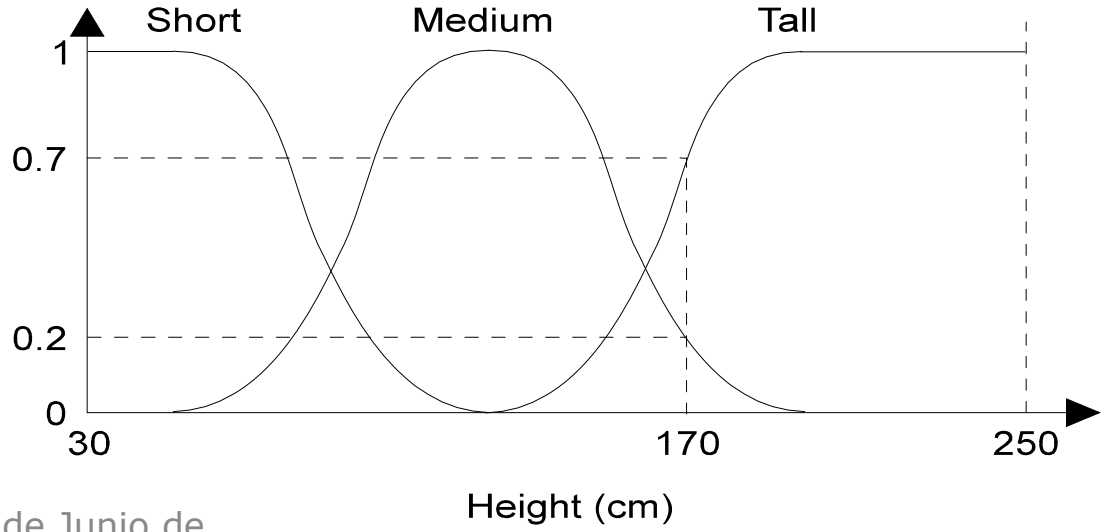
- ✓ Introduction to Soft Computing/Computational Intelligence
- ✓ Fuzzy Sets and Fuzzy Systems
- ✓ Genetic Algorithms
- ✓ Neural Netwoks
- ✓ Concluding Remarks

Fuzzy Sets and Fuzzy Systems



Fuzzy Sets - 1965 Lotfi Zadeh, Berkely

Fuzzy sets are sets whose elements have degrees of membership, as an extension of the classical notion of set.



Fuzzy Sets and Fuzzy Systems

Example:

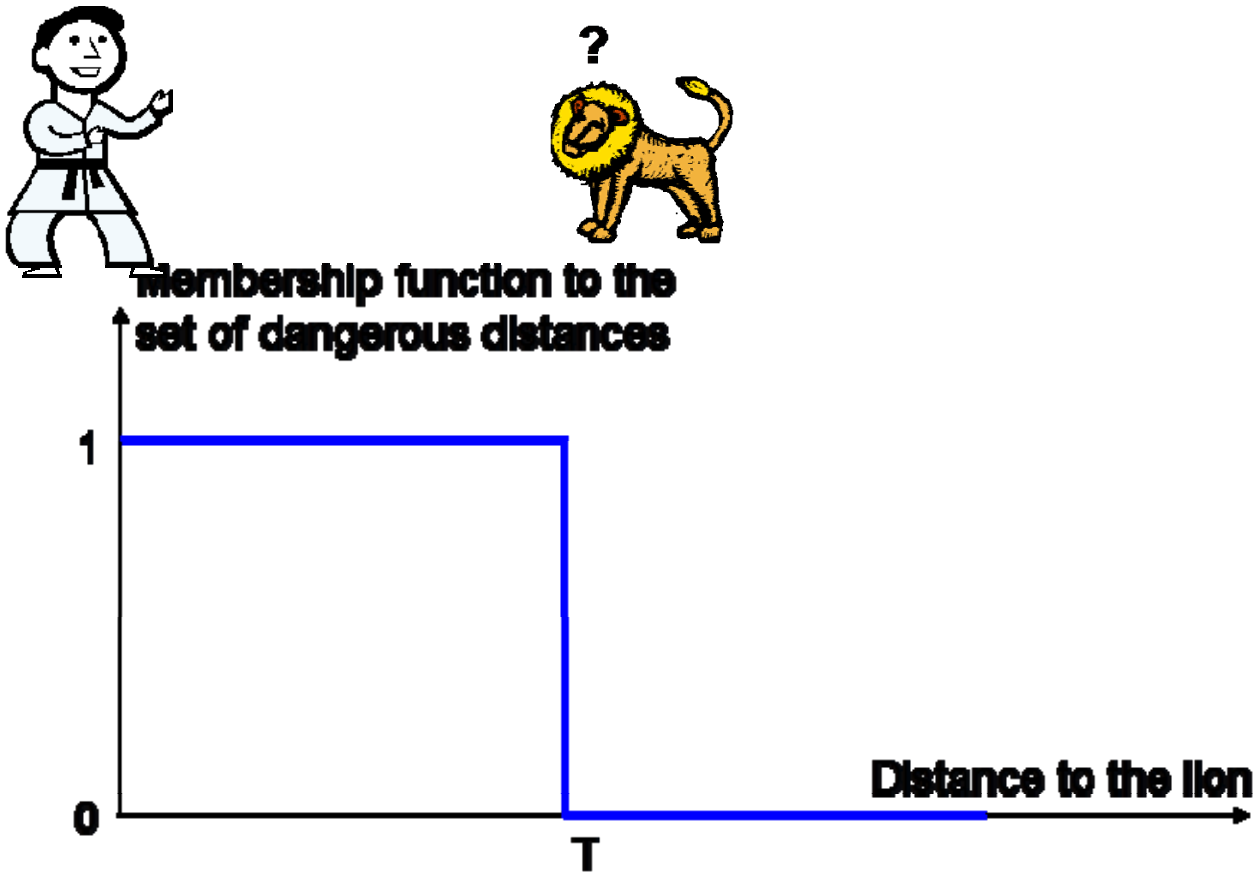
"It is dangerous to come too close to the lion"

Fuzzy Sets and Fuzzy Systems

Translating the above sentence means choosing a threshold T :

If (distance < T) then danger

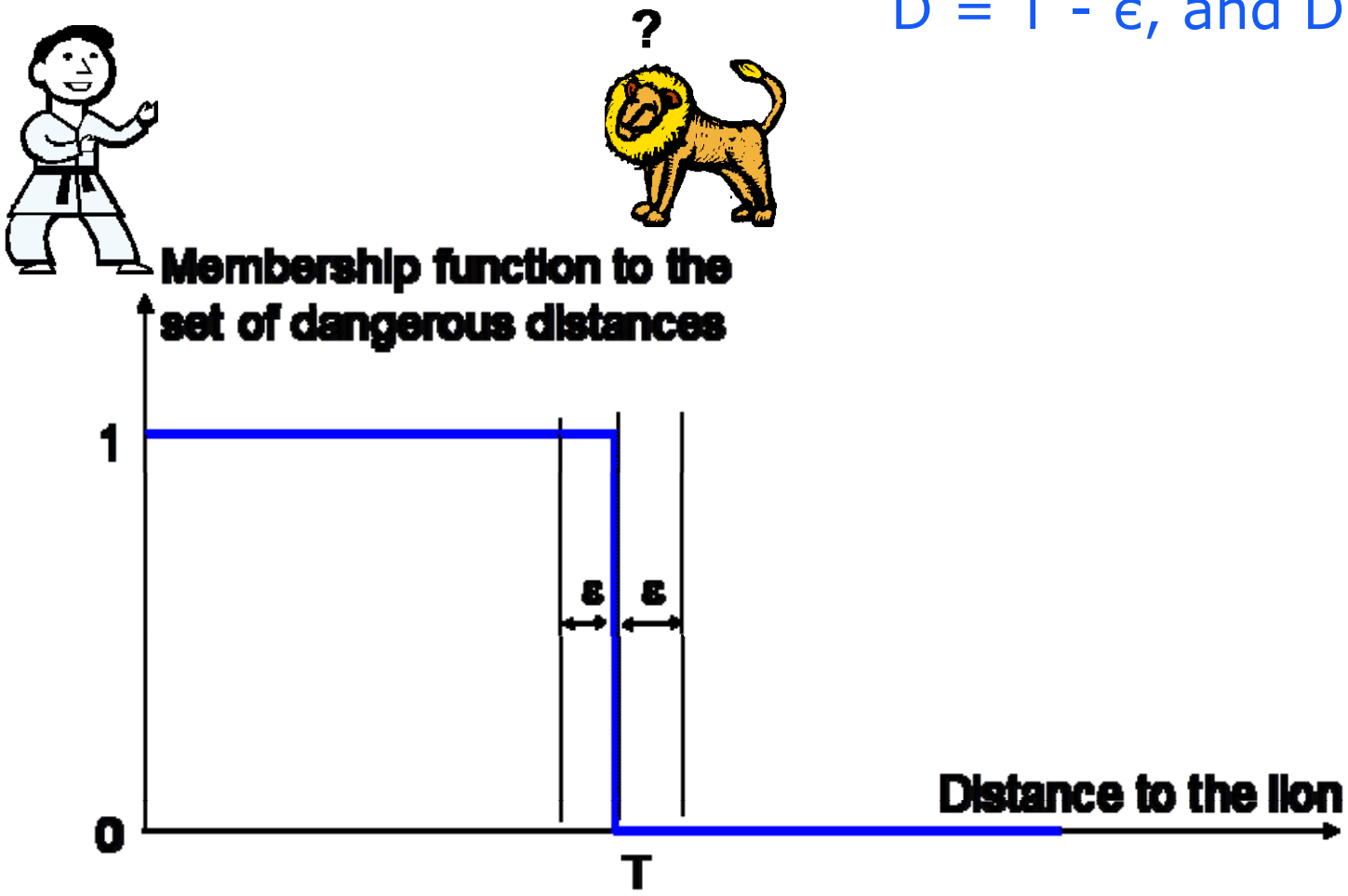
One can represent this threshold with the classical sets theory :



Fuzzy Sets and Fuzzy Systems

Let us consider a very small distance ϵ :
then one can see that the membership function varies from 1 to 0 for the 2 very close situations

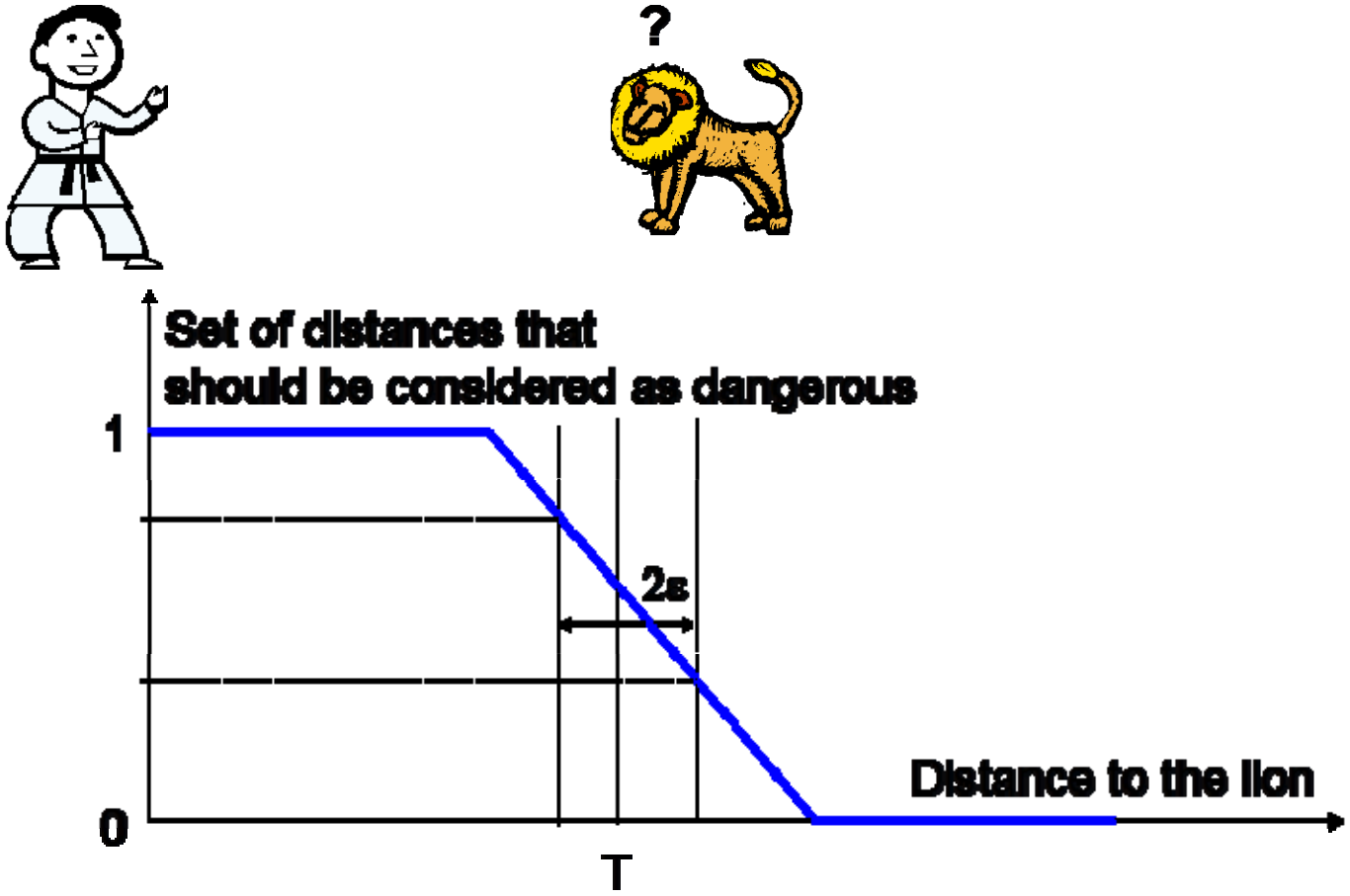
D = T - ϵ, and D = T + ϵ :



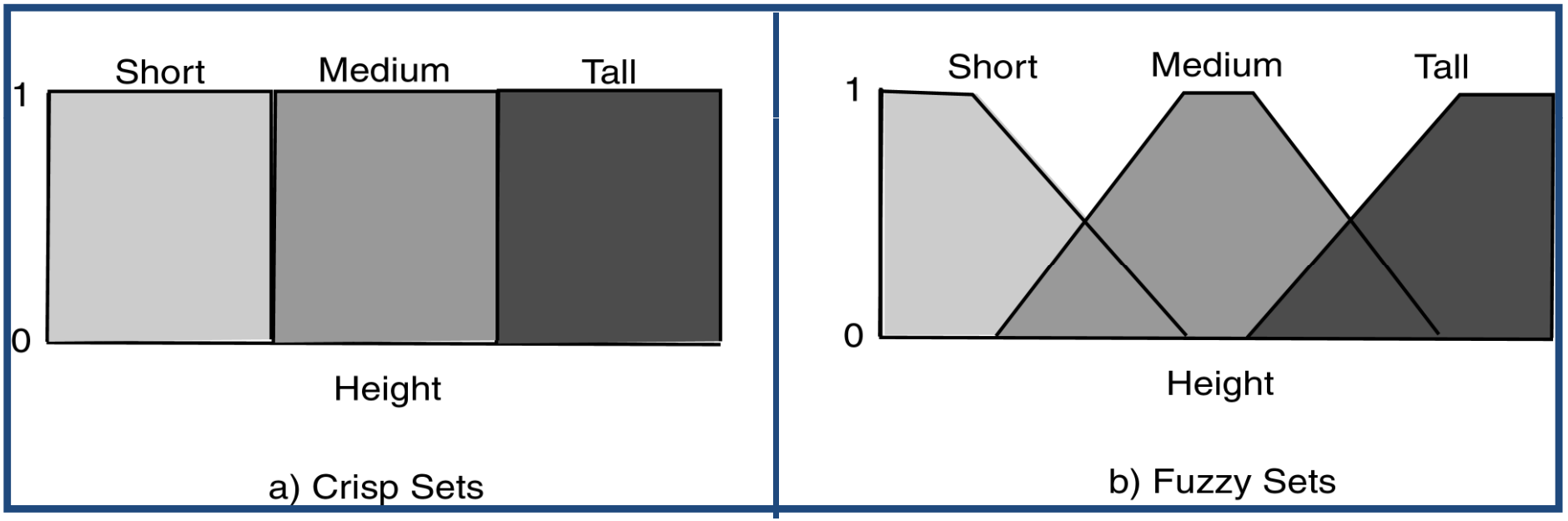
Fuzzy Sets and Fuzzy Systems

Continuous membership functions : fuzzy sets

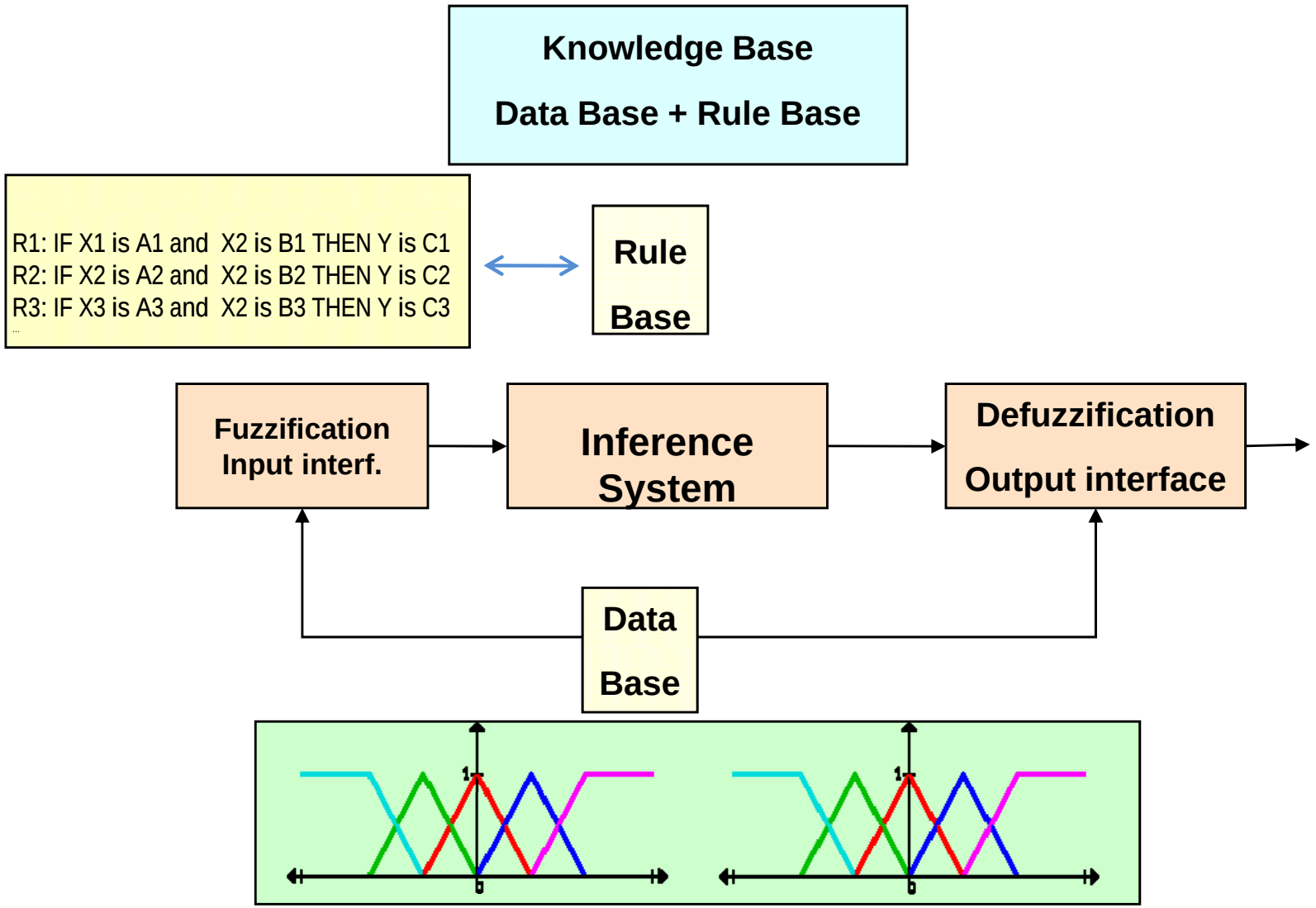
The idea is to use a continuous function as membership function :



Fuzzy Sets


Short
Medium
Tall
1
0
Height
a) Crisp Sets
Short
Medium
Tall
1
0
Height
b) Fuzzy Sets

Fuzzy Sets and Fuzzy Systems

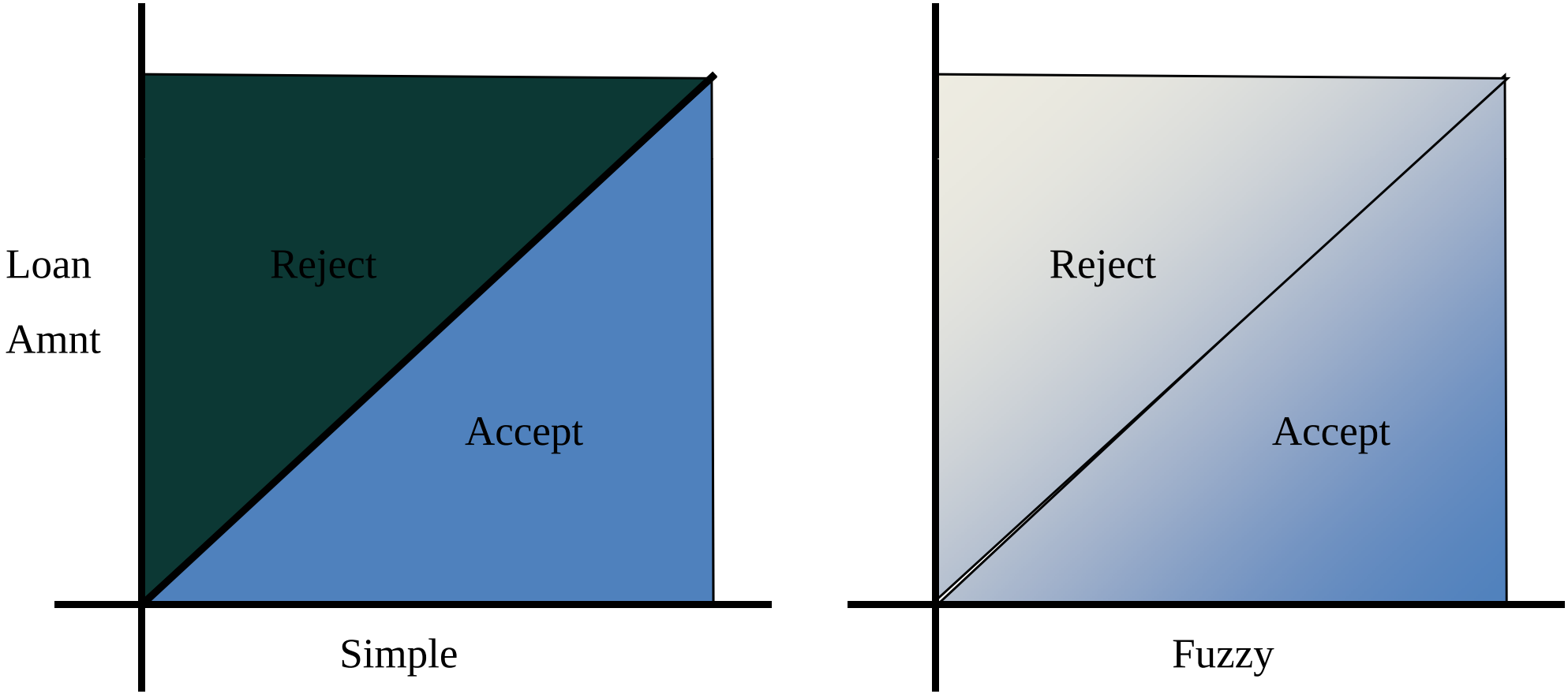


Fuzzy rule-based system / Fuzzy logic controller (FLC)

Fuzzy Sets and Fuzzy Systems

Classification/Prediction is Fuzzy

DM: Prediction and classification are fuzzy.



Outline

- ✓ Introduction to Soft Computing/Computational Intelligence
- ✓ Fuzzy Sets and Fuzzy Systems
- ✓ Genetic Algorithms
- ✓ Neural Netwoks
- ✓ Concluding Remarks

Genetic Algorithms

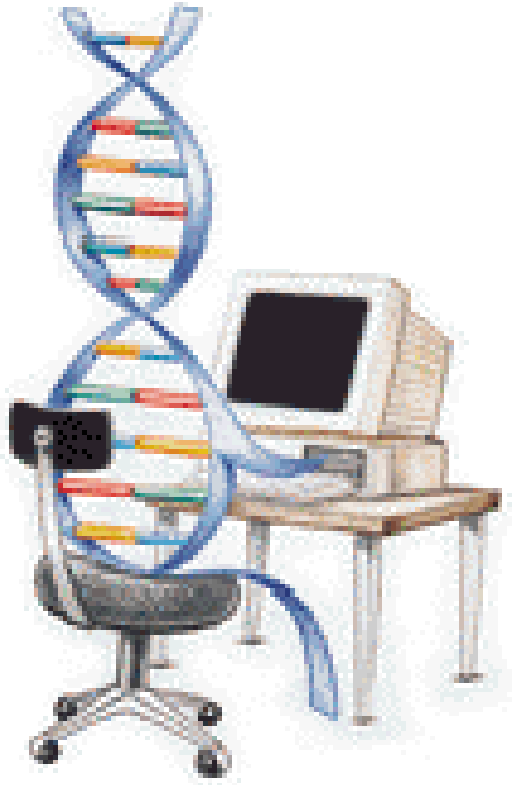
1. **GENETIC ALGORITHMS. INTRODUCTION**
2. **HOW TO CONSTRUCT THEM?**
3. **ON THE USE OF GENETIC ALGORITHMS**
4. **MODELS: GENERATIONAL VERSUS STEADY STATE**
5. **APPLICATIONS**
6. **EXAMPLE: TSP**
7. **SOFTWARE AND IMPLEMENTATIONS**
8. **CONCLUDING REMARKS**

1. GENETIC ALGORITHMS. INTRODUCTION

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

What is a genetic algorithm?

Genetic algorithms



They are optimization algorithms,

search

and learning

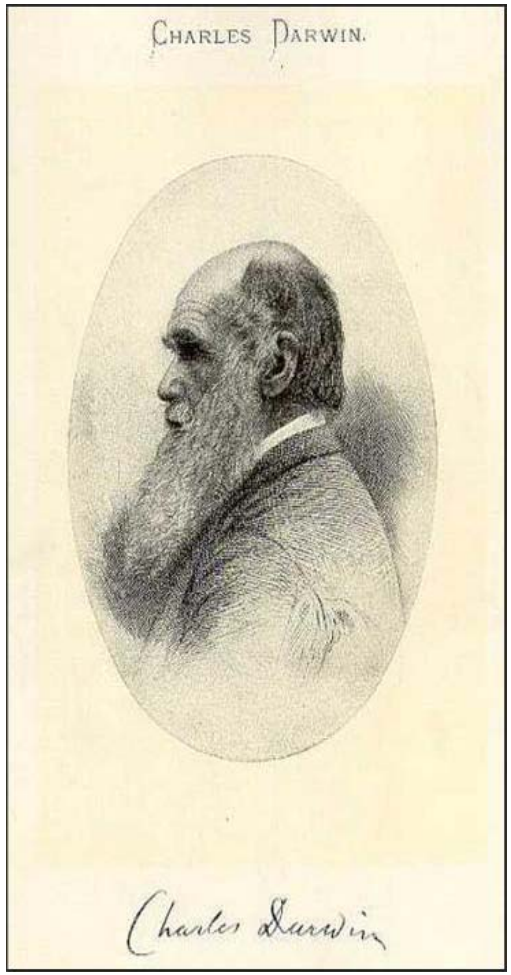
inspired in the process of

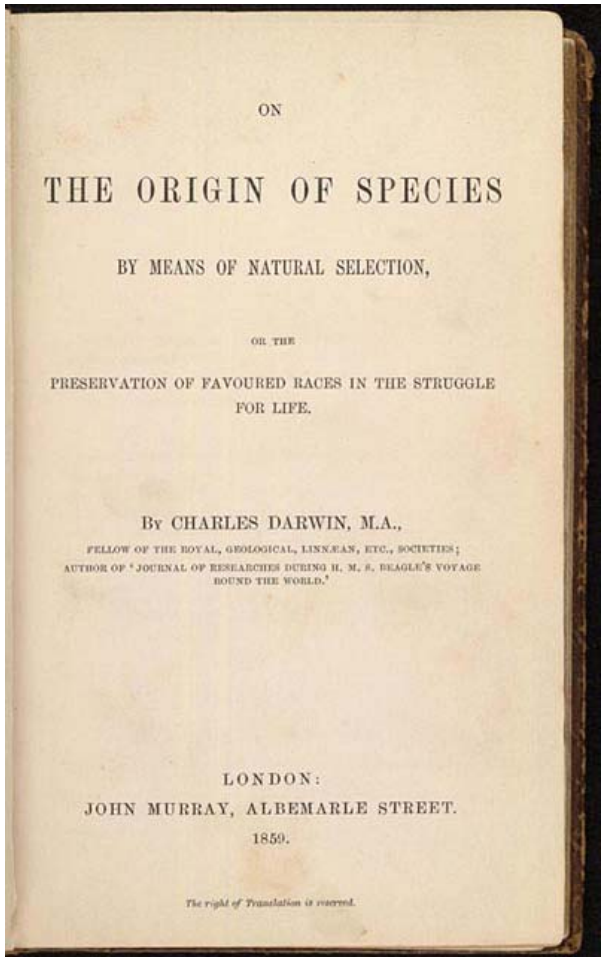
Natural and Genetic

Evolution

What is a genetic algorithm?

Natural Evolution



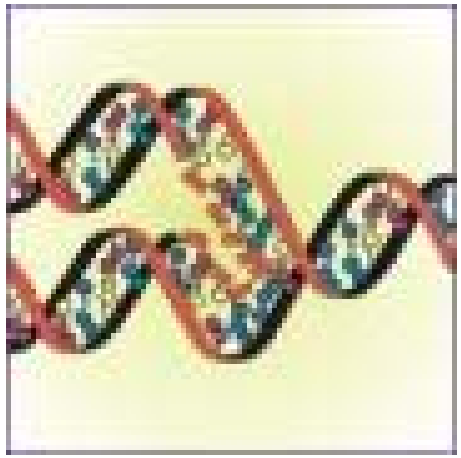


What is a genetic algorithm?

Artificial Evolution

EVOLUTIONARY COMPUTATION

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

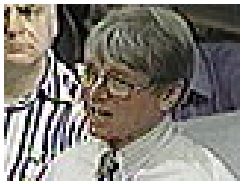


What is a genetic algorithm?

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 & Evolutiontechnique
Technical University Berlin
http://www.bionik.tu-berlin.de/

Evolutionary Programming. 1960-1966, Florida

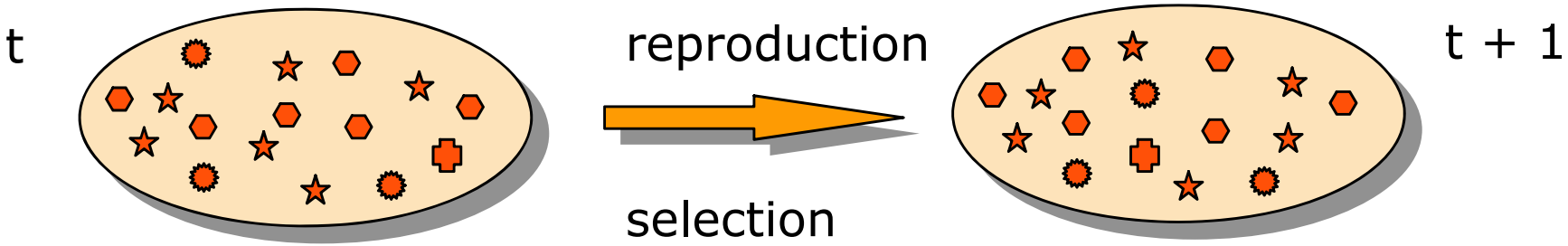
Lawrence J. Fogel,
Natural Selection, Inc.
Inventor of Evolutionary Programming

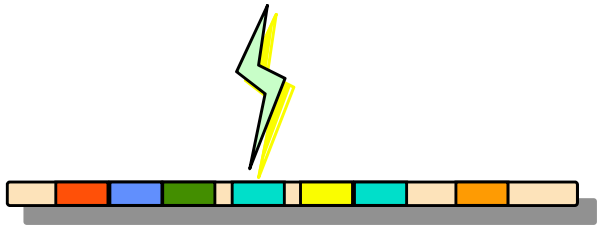
Genetic Programming.1989, Stanford University

John Koza
Stanford University.
Inventor of Genetic Programming

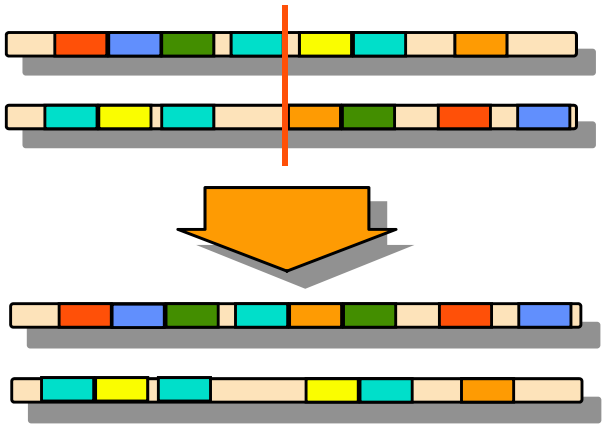
Thre exist other modelos based on population evolution

The ingredients



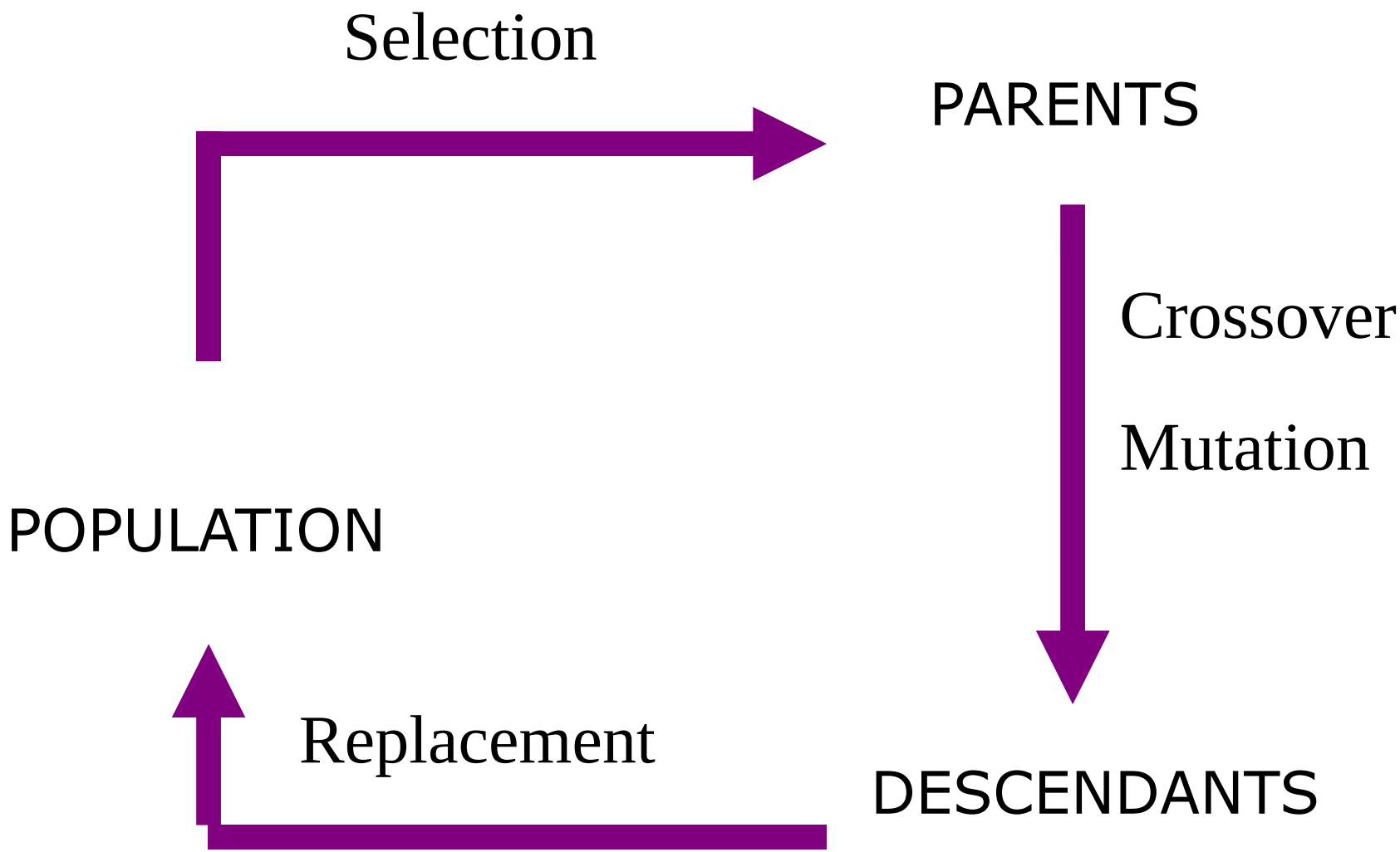


mutation



Crossover
(or recombination)

The evolution cycle



Genetic Algorithm Structure

Basic Genetic Algorithms

Beginning (1)

t = 0

Initialization P(t)

evalution P(t)

While (the stop contition is not verified) do

Beginning (2)

t = t + 1

selection P’(t) from P(t-1)

P’’(t) ← crossover P’(t)

P’’’(t) ← mutation P’’(t)

P(t) ← 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**

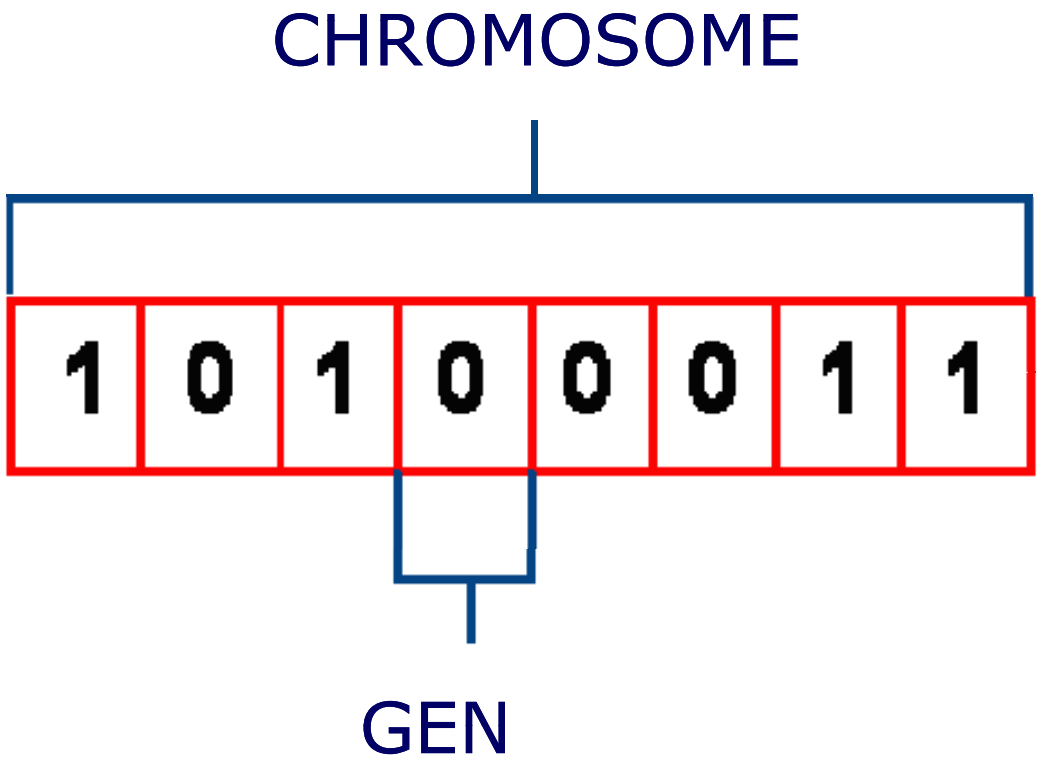
PROBLEM DEPENDENT

ALGORITHM COMPONENTS

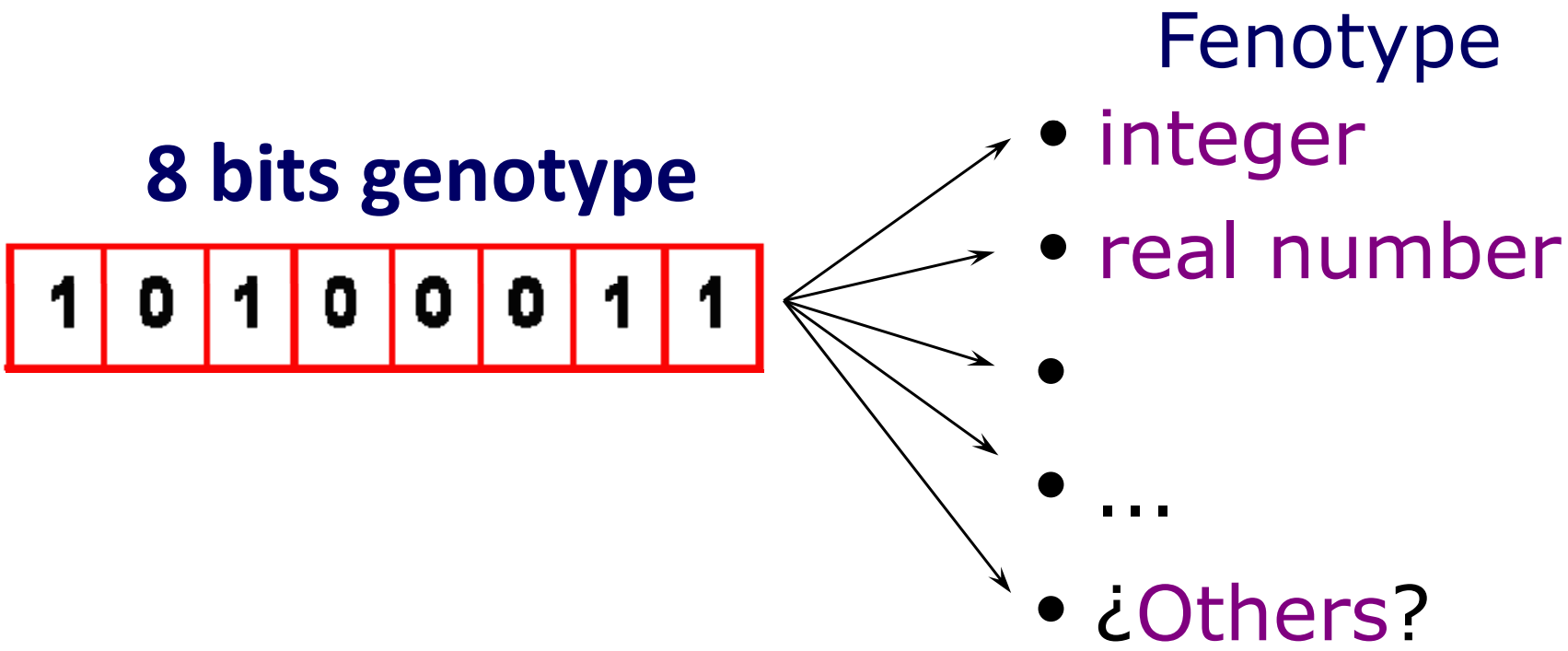
Representación

- 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.

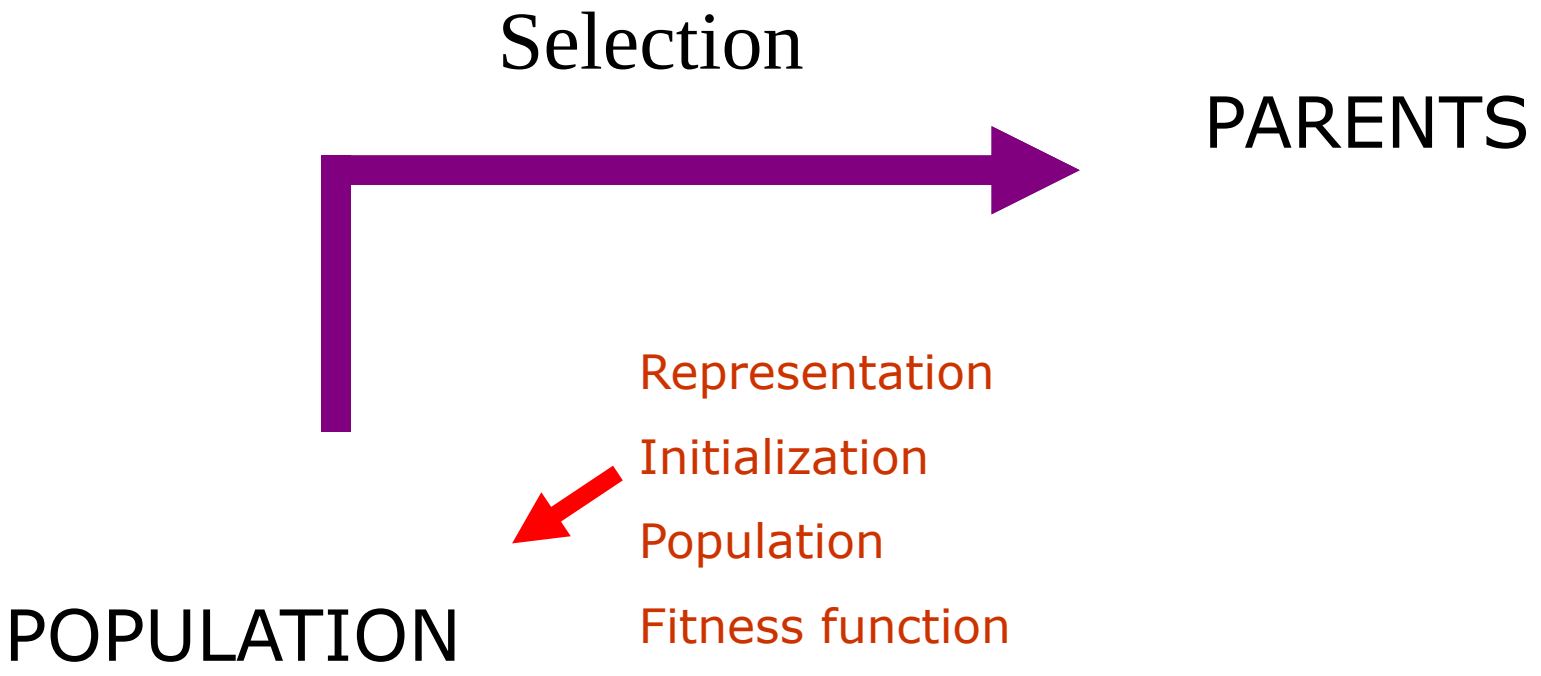
Initialization

- 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.
- Subrutine, 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.

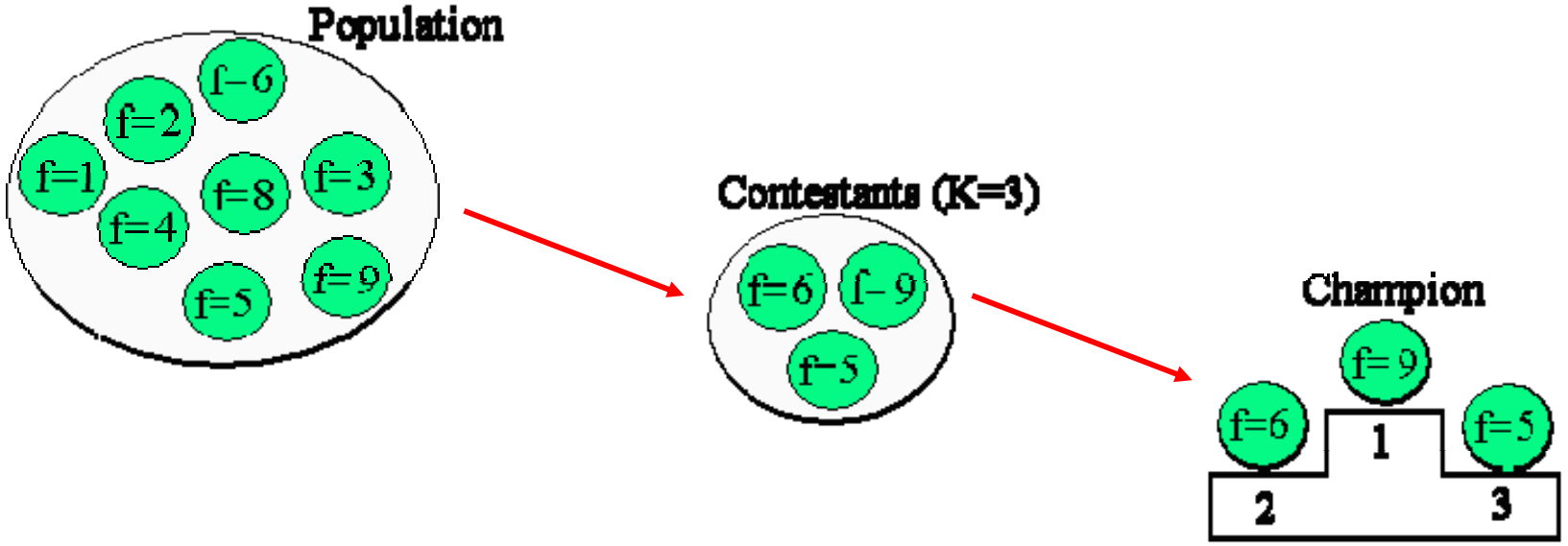
Strategy of selection

Tournament selection

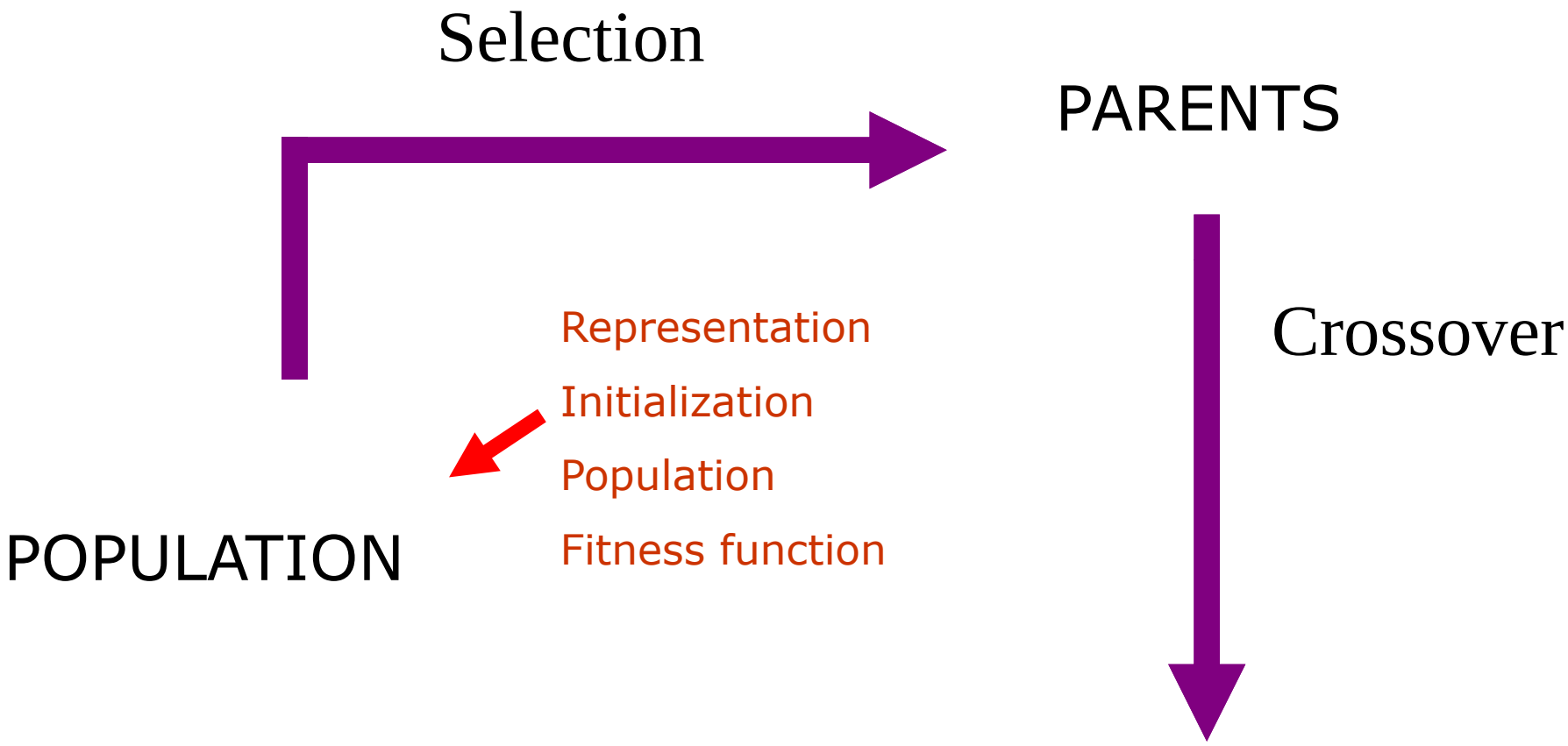
For each parent:

- 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?



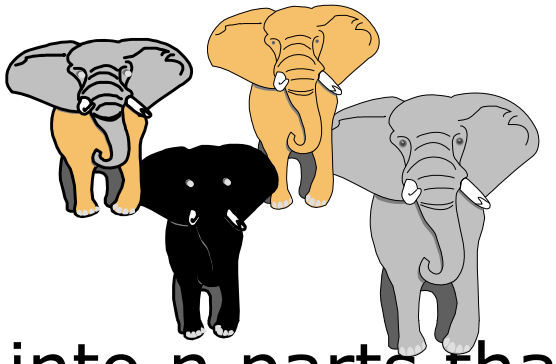
Crossover operator

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 (Pc between 0.6 and 0.9, usually).

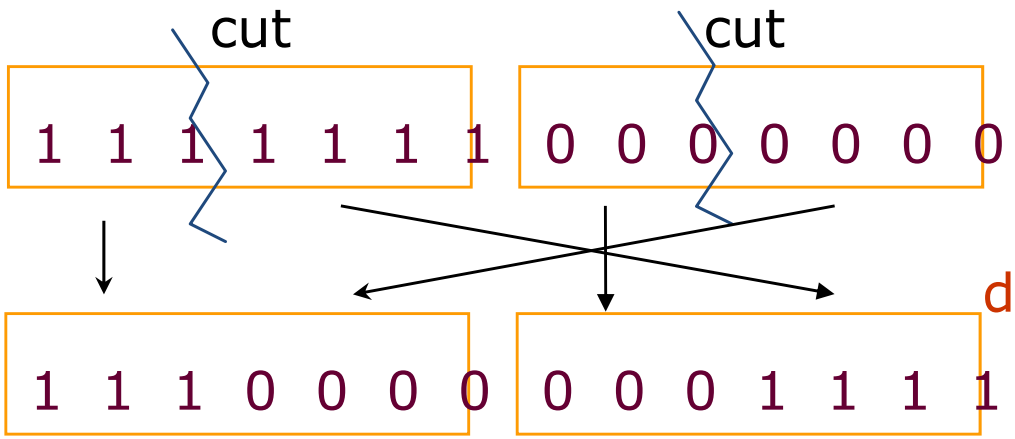
Example: Simple crosover on the binary representation

Population:

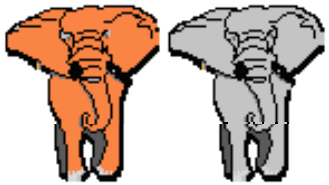


. . .

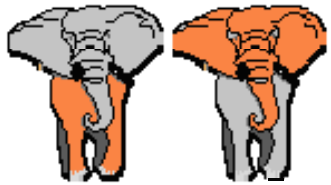
Each chromosome is divided into n parts that are recombined (example for n = 2)



parents

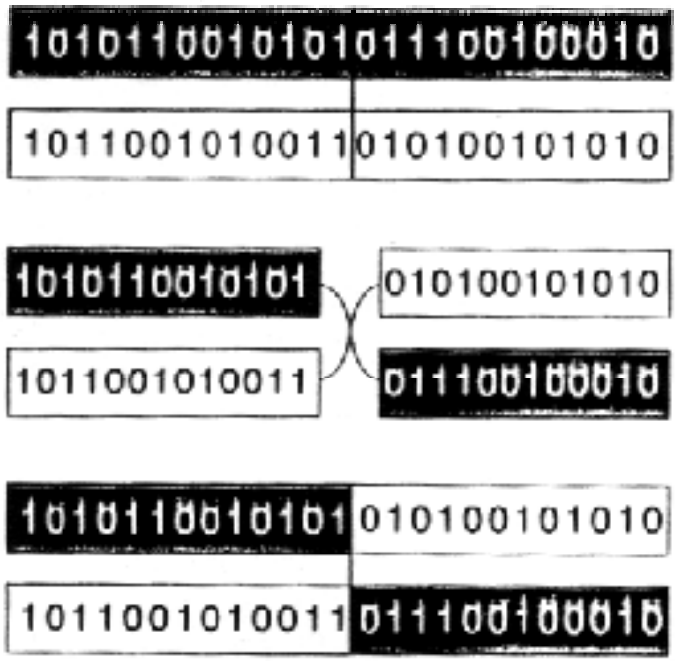


descendants



Crossover operator

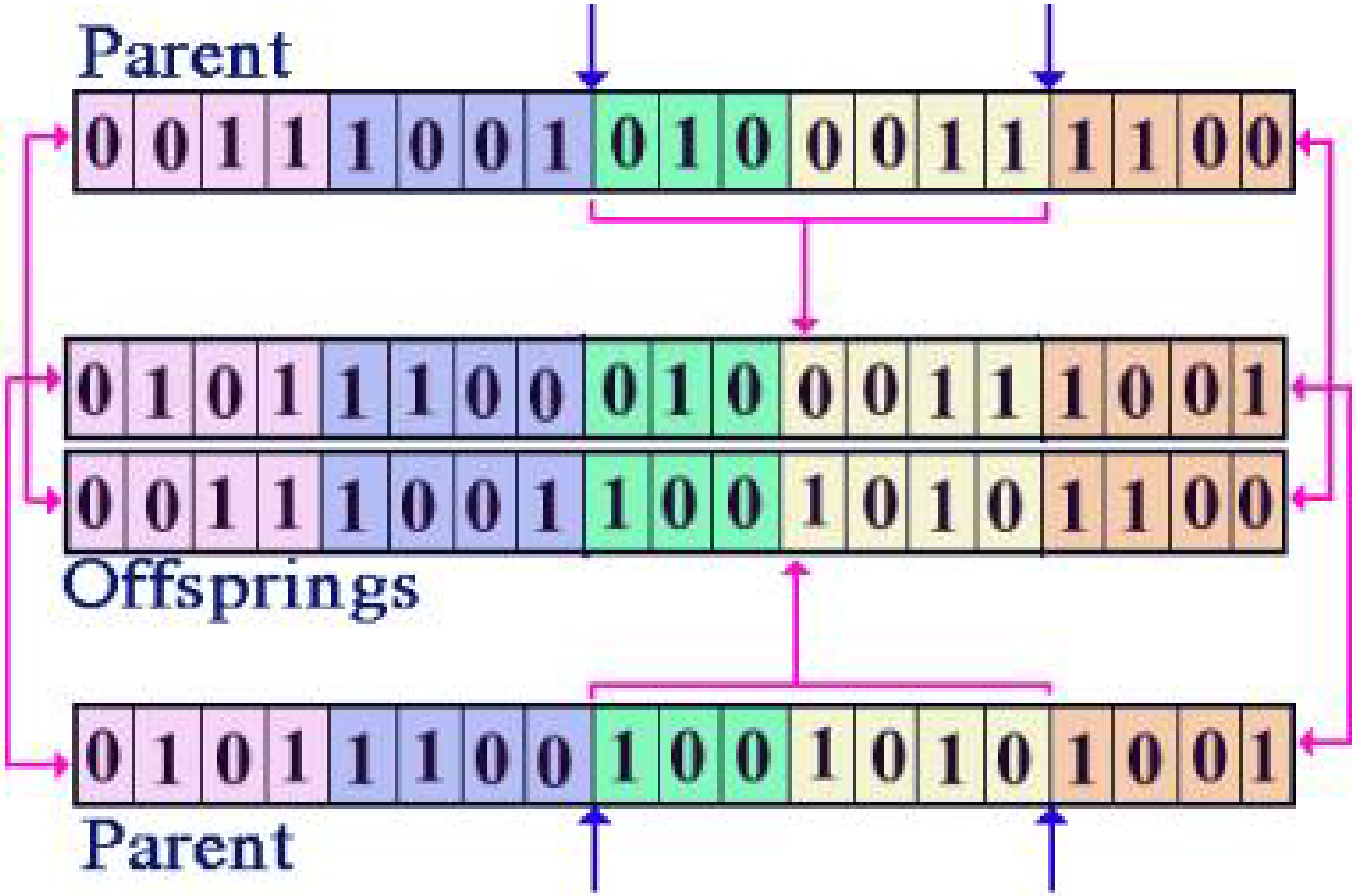
Classical image (John Holland): Biologica crosover



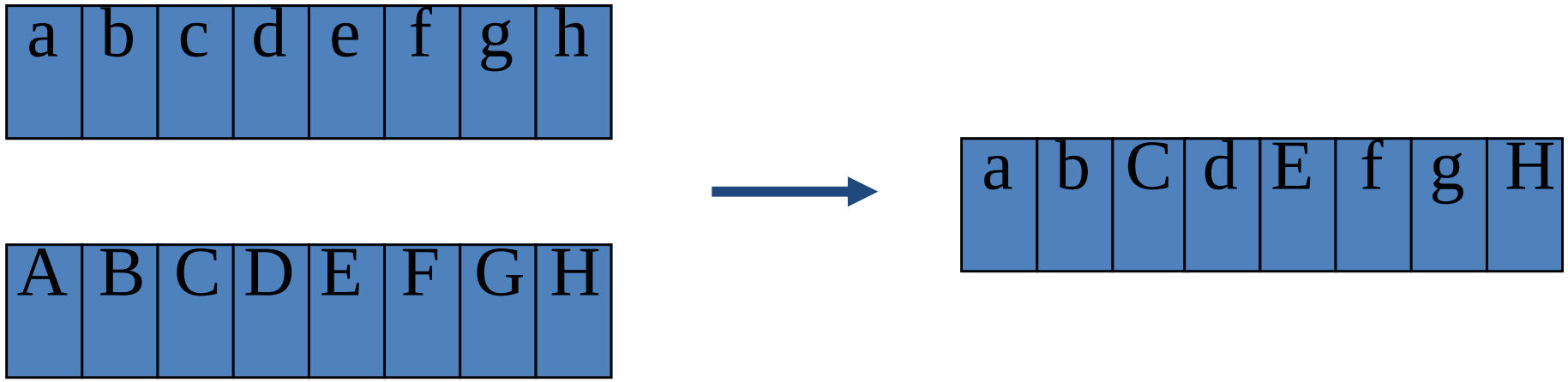
CROSSOVER is the fundamental mechanism of genetic rearrangement for both real organisms and genetic algorithms.

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

Example: Two points crossover

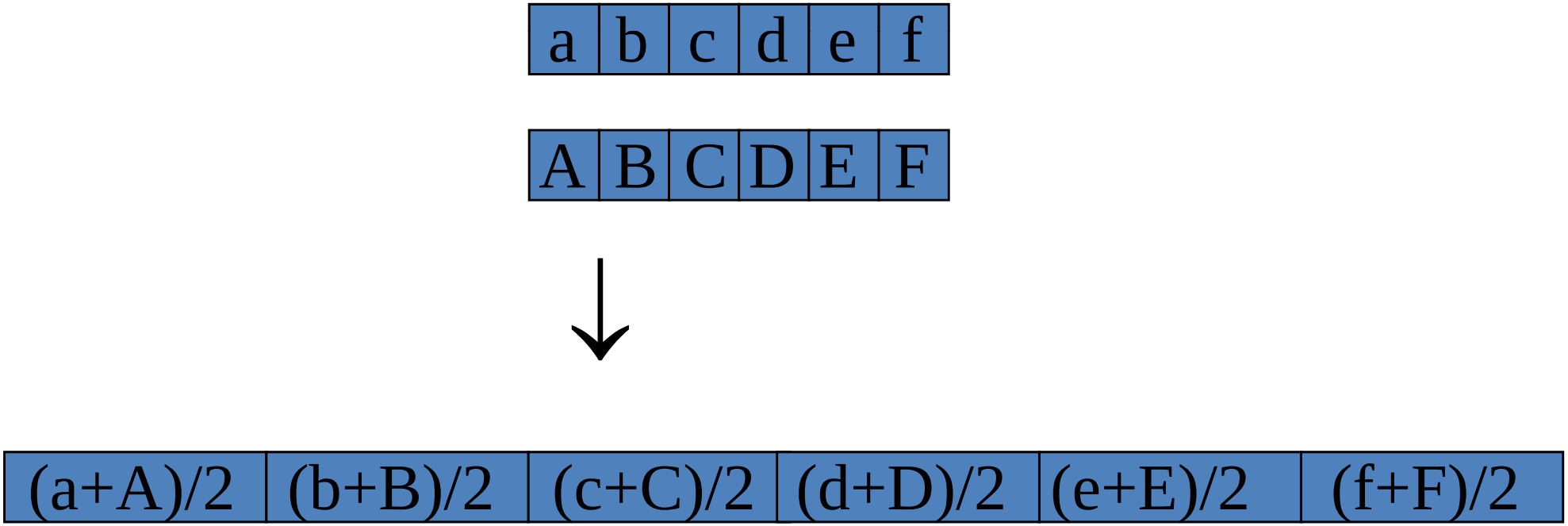


Example: uniform crossover



Example: Real coding crossover operator

Arithmetic crossover:



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

- Two chromosomes

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

- BLX- α generates two descendants

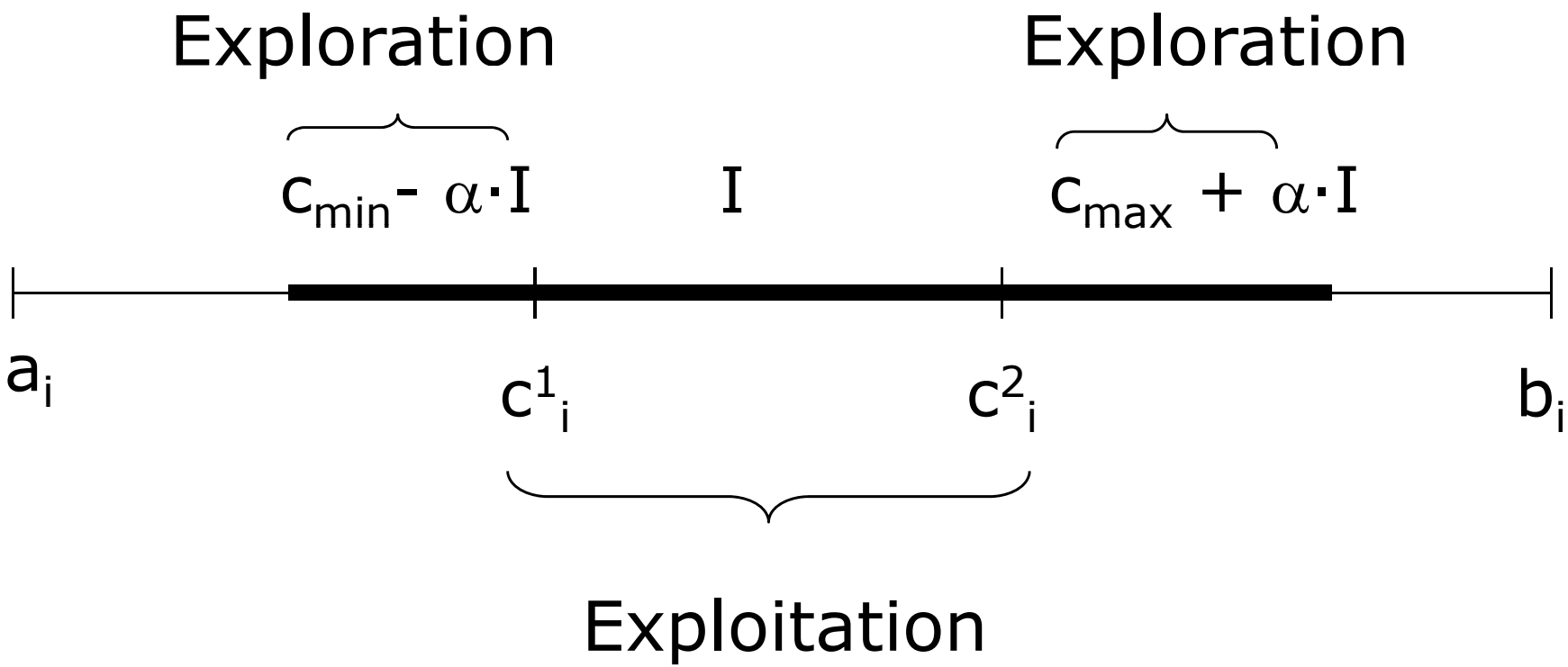
$$H_k = (h_{k1},\ldots, h_{ki},\ldots, h_{kn}) , k = 1,2$$

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

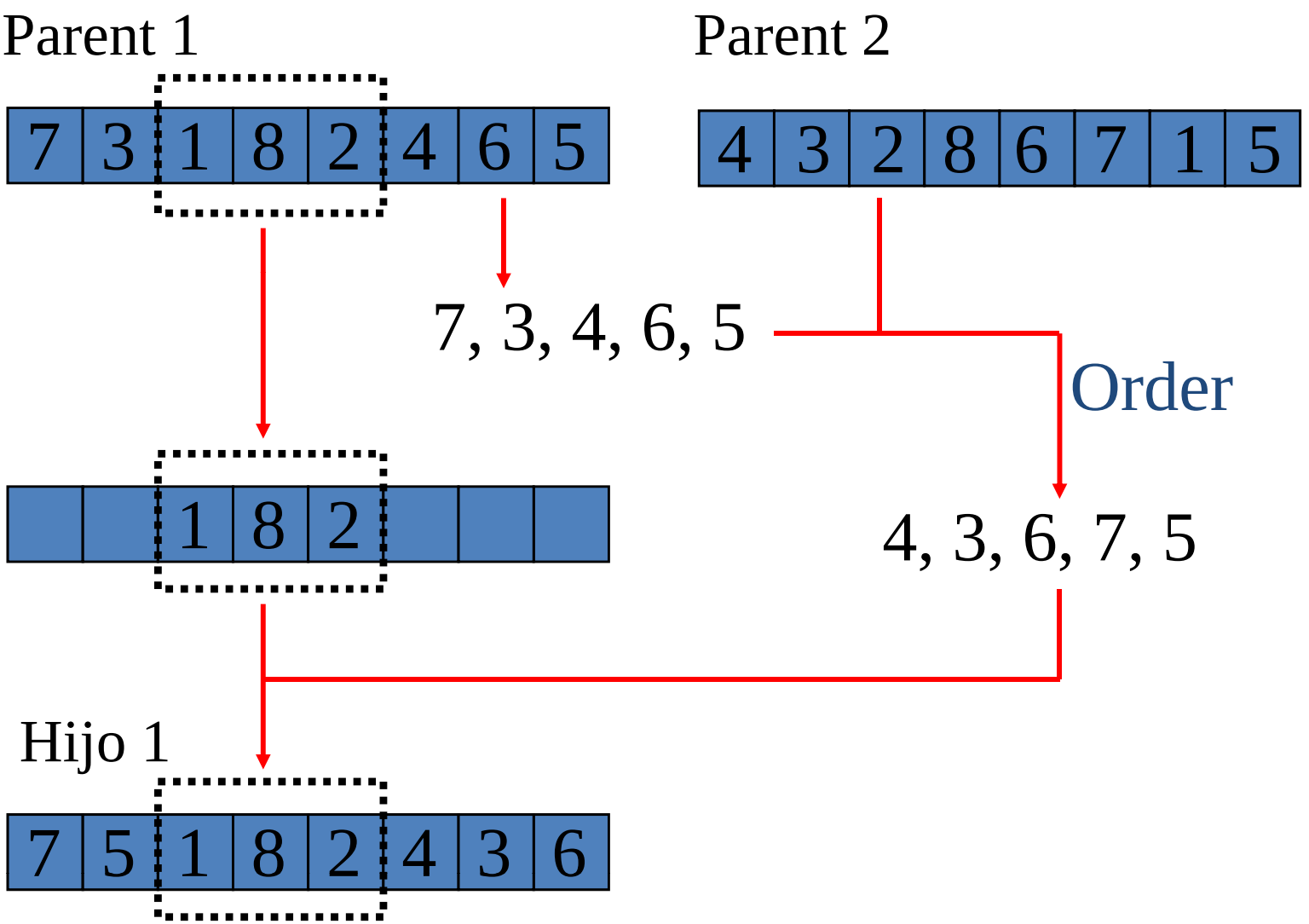
$$[C_{min} - I\cdot\alpha, C_{max} + I\cdot\alpha]$$

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

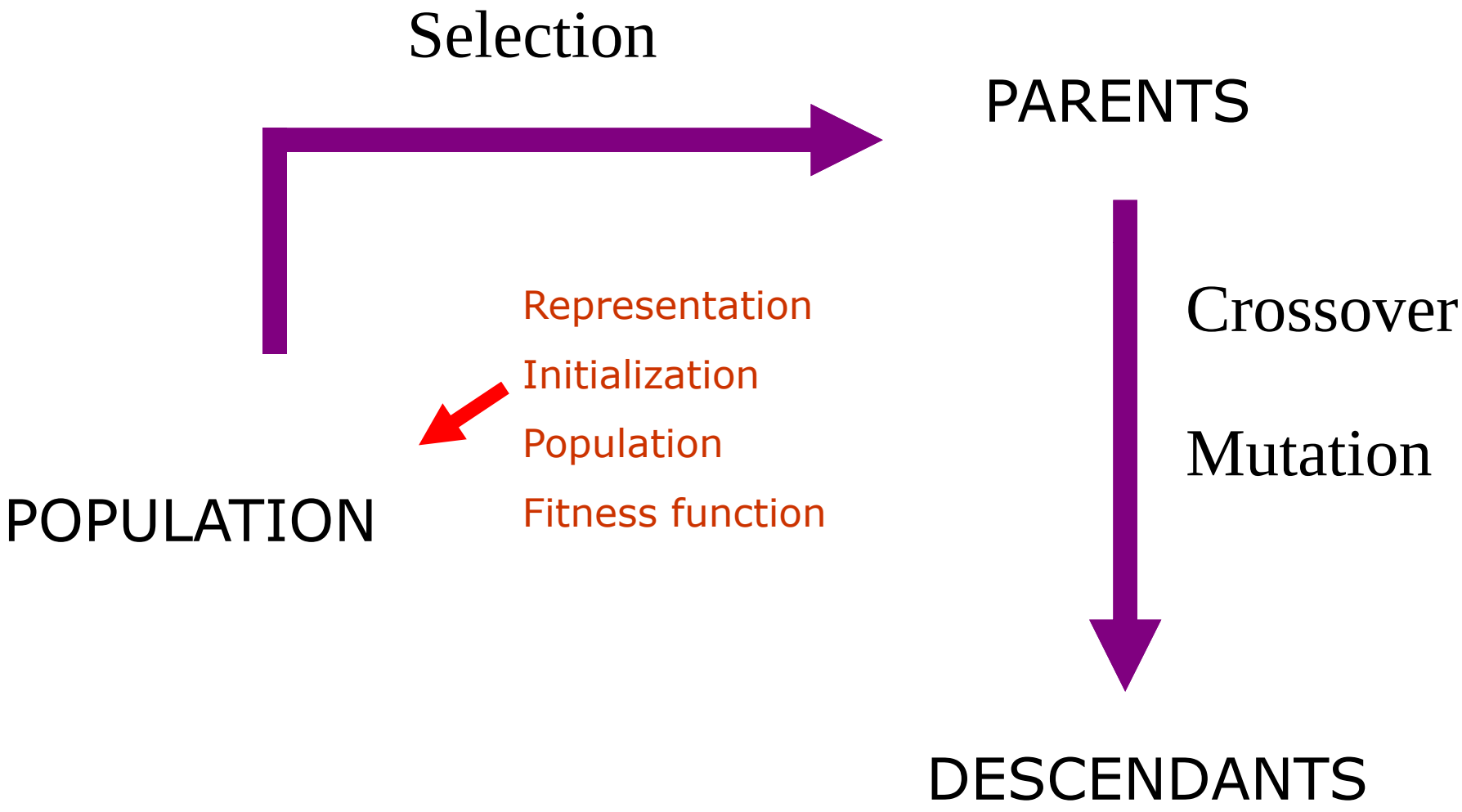
Ejemplo: Operador de cruce para representación real: BLX-α



Example: Crossover operator for order representation: OX



HOW TO CONSTRUCT A GA?



Mutation operator

Features:

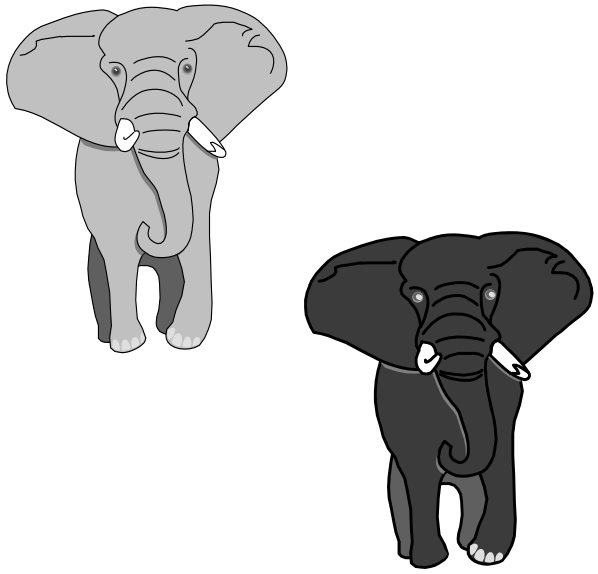
- It must allow us to reach any point through a sequence of runs.
- We must control the size.
- It must produce valid chromosomes.
- It is used with a low running probability on the descendant obtained after the application of the crossover operator.

Example: binary mutation

before 1 1 1 1 1 1 1

after 1 1 1 0 1 1 1

Mutated gen



The mutation happens with a low running probability per gen p_m

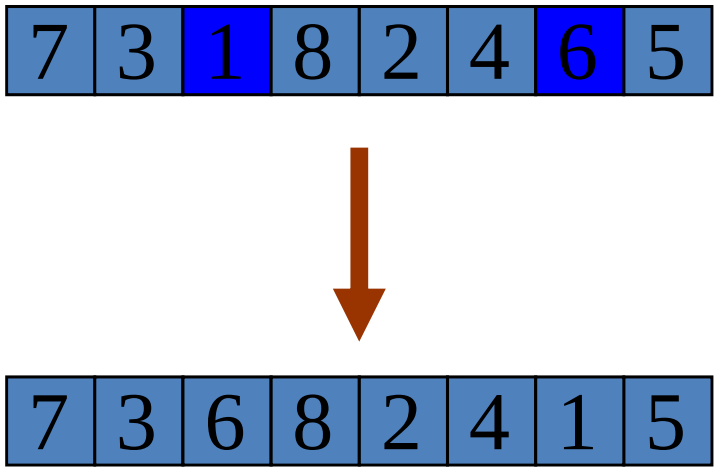
Example: real coding mutation

- 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

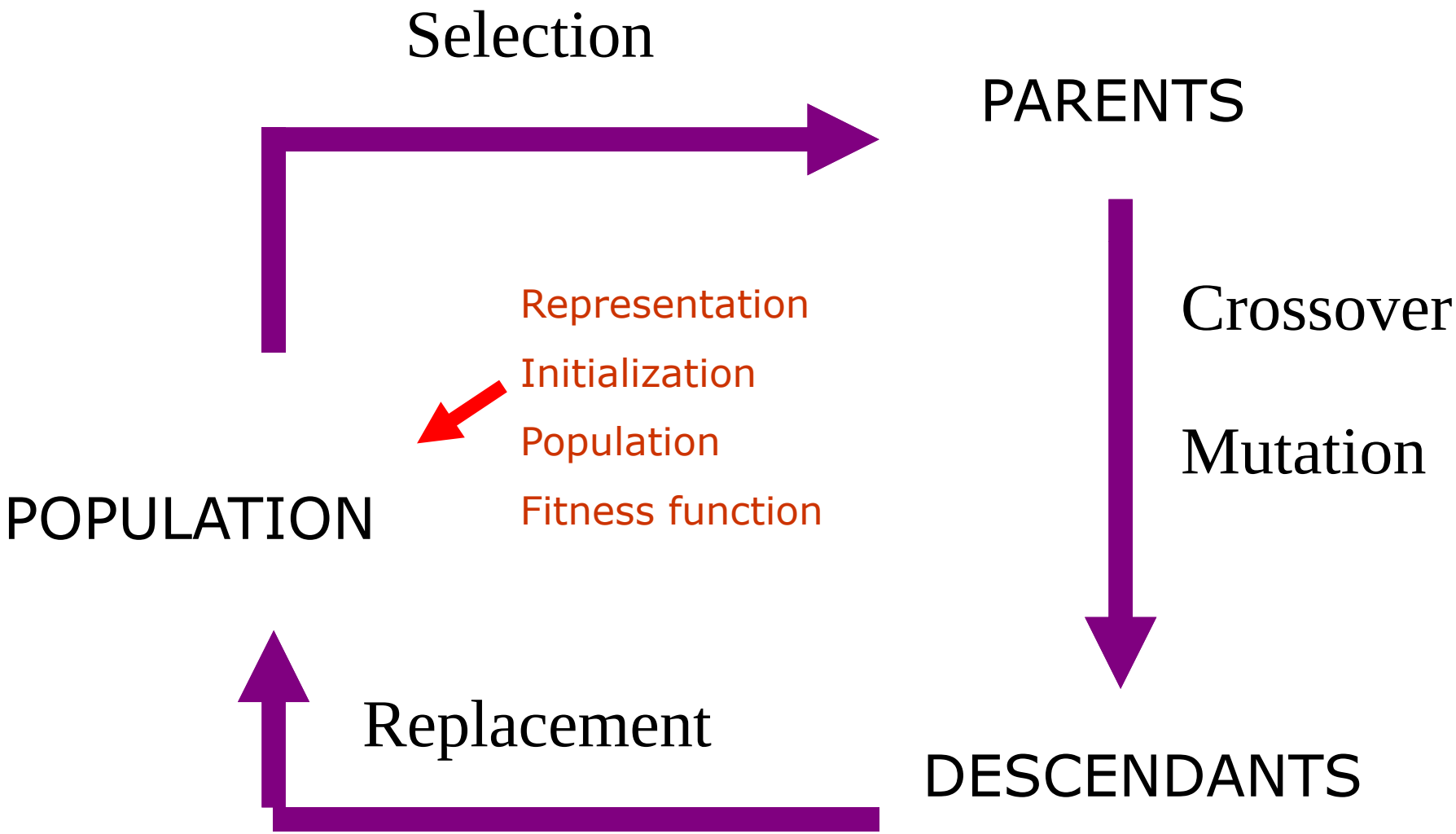
$x'_i = x_i + N(0,\sigma_i)$

For each parameter.

Example: order representation mutation



HOW TO CONSTRUCT A GA?



Replacement strategy

Complete population

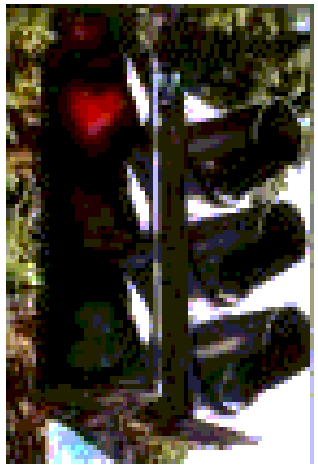
Elitism: Maintaining the best chromosome

Replacement of parents per children (via competition)

....

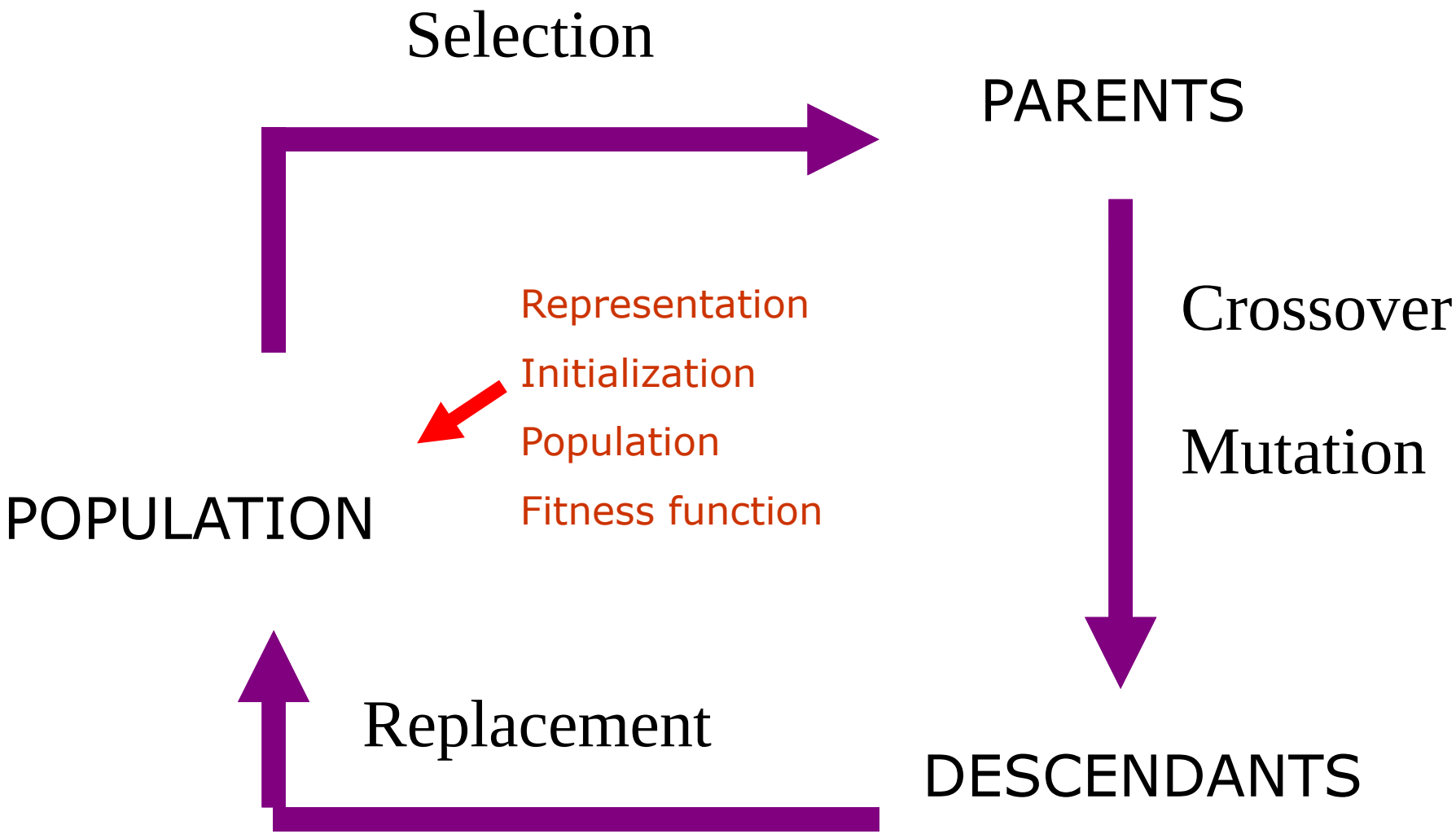
It depends on the model.

Stop condition



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

Components


Selection
PARENTS
Crossover
Mutation
DESCENDANTS
Replacement
POPULATION
Representation
Initialization
Population
Fitness function

3. MODELS: GENERATIONAL vs STEADY STATE

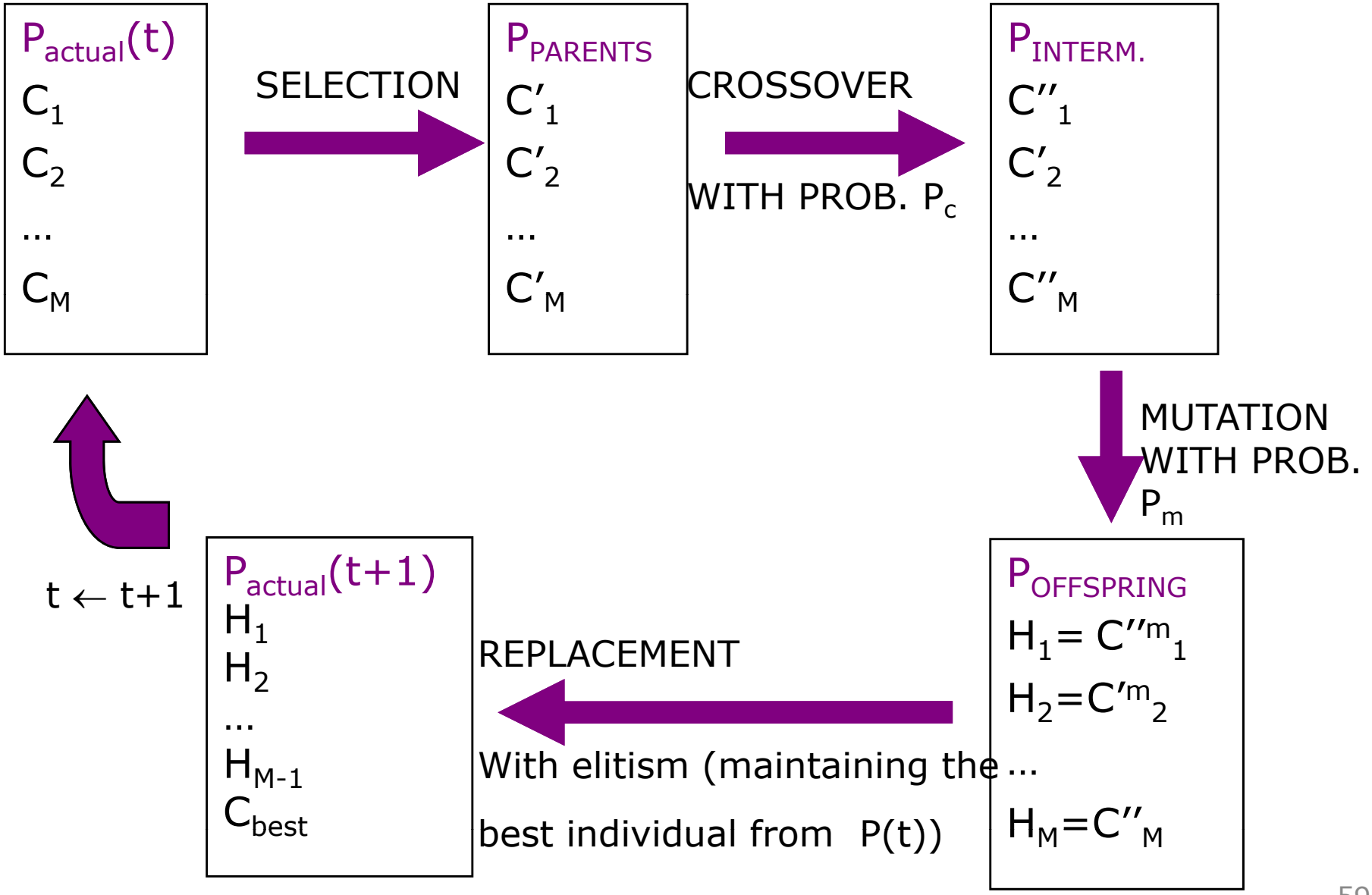
Generational model: Replacement of the complete population

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

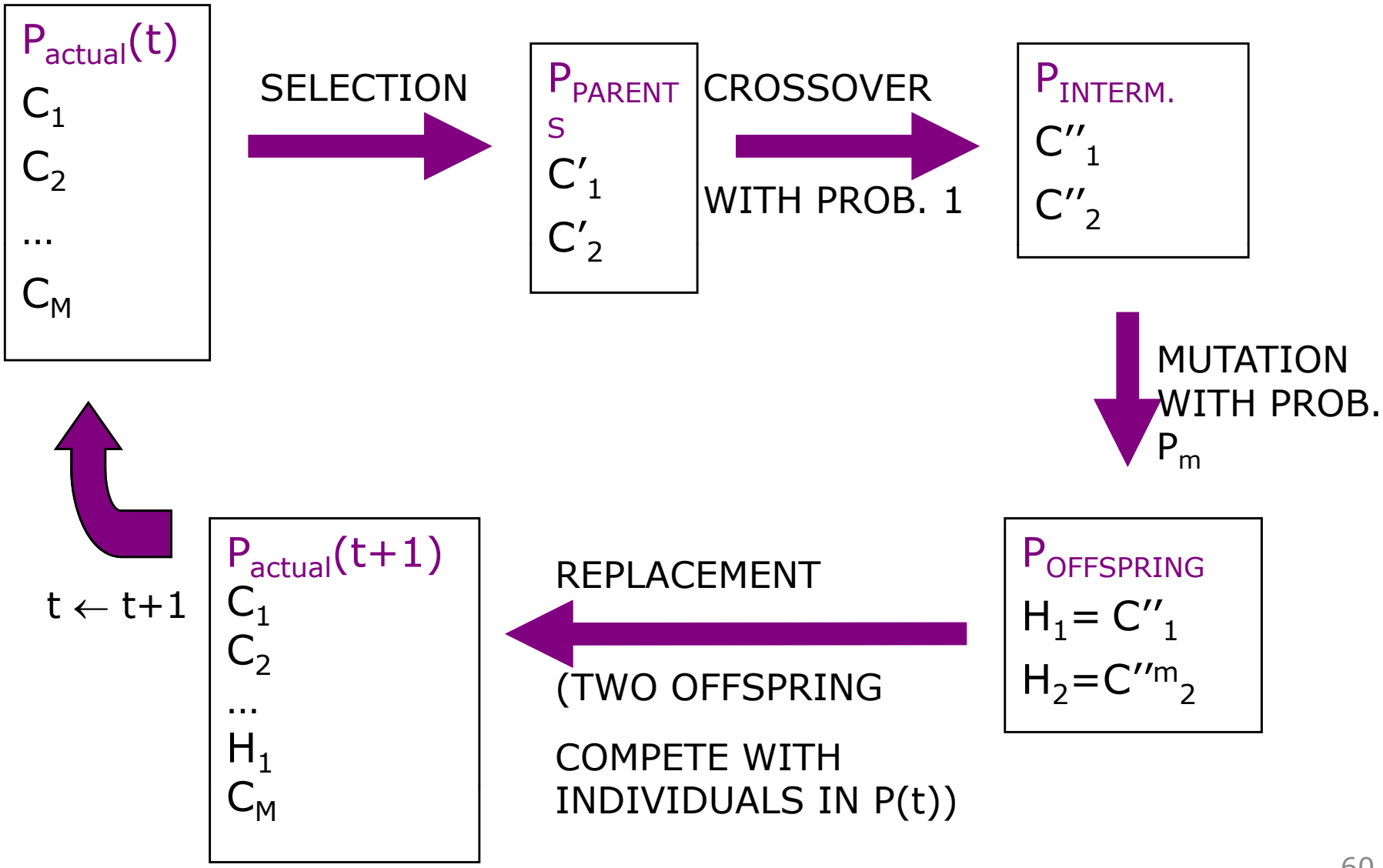
Only one/two individuals are replaced in the population.

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

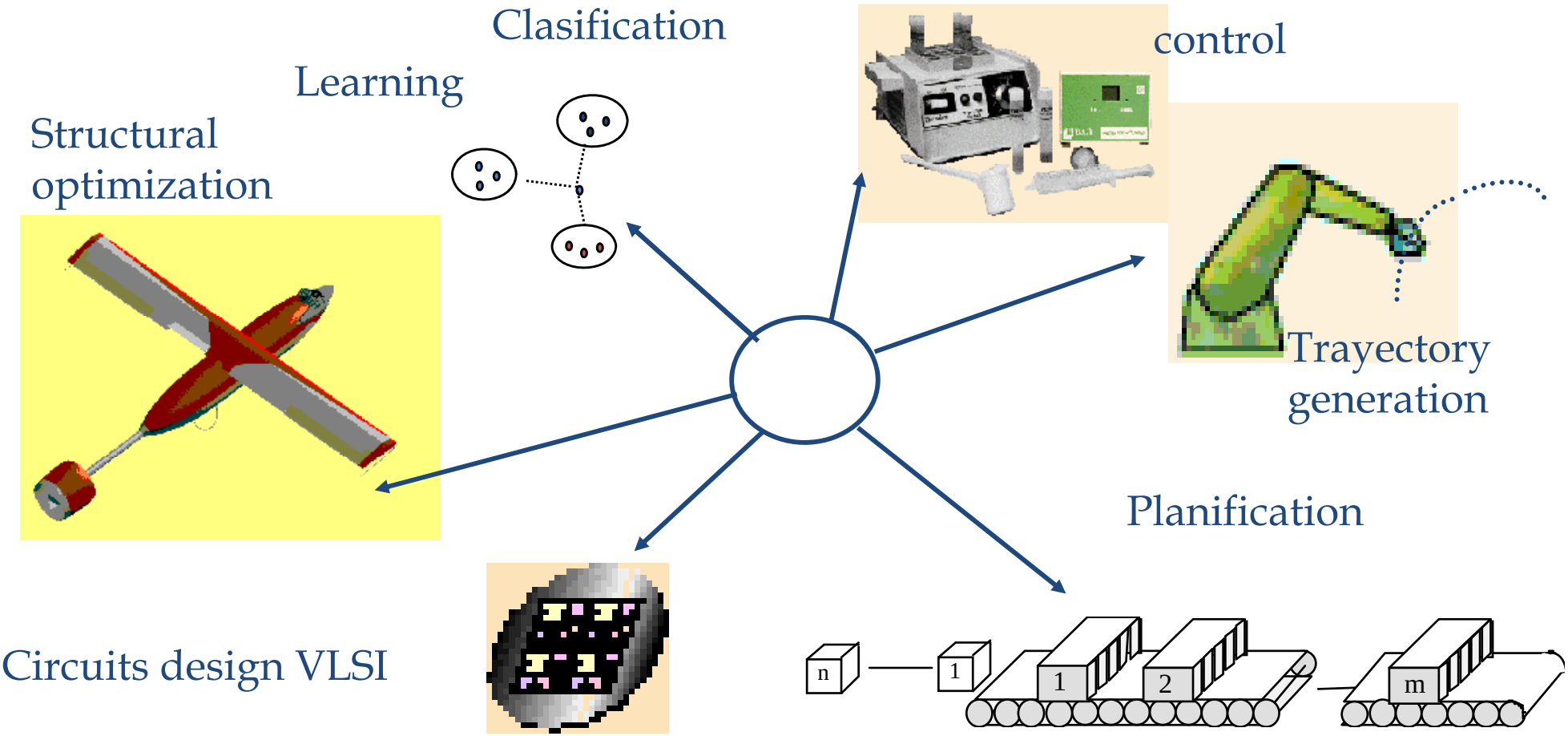
Generational model



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

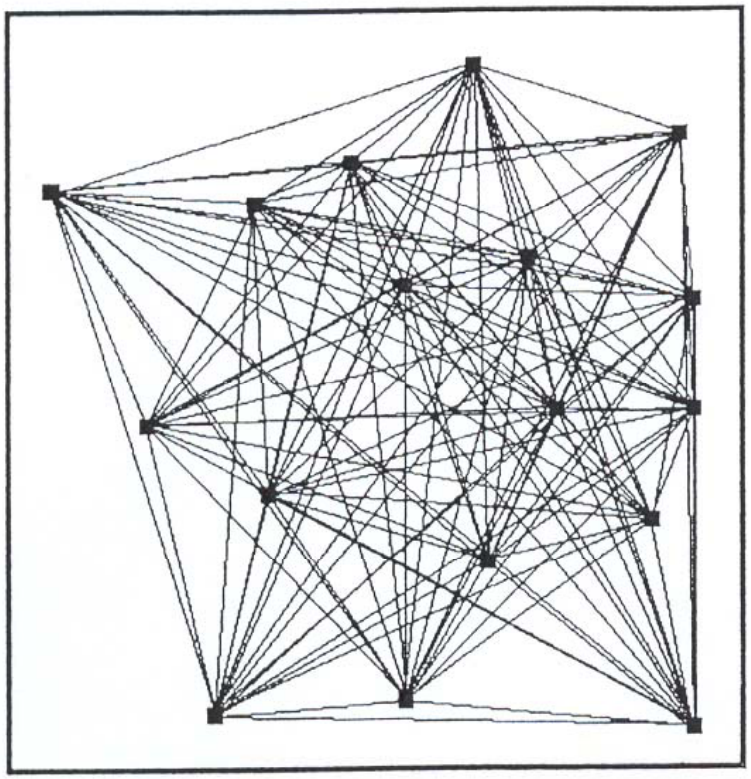
Objective: Sum of distance among cities.

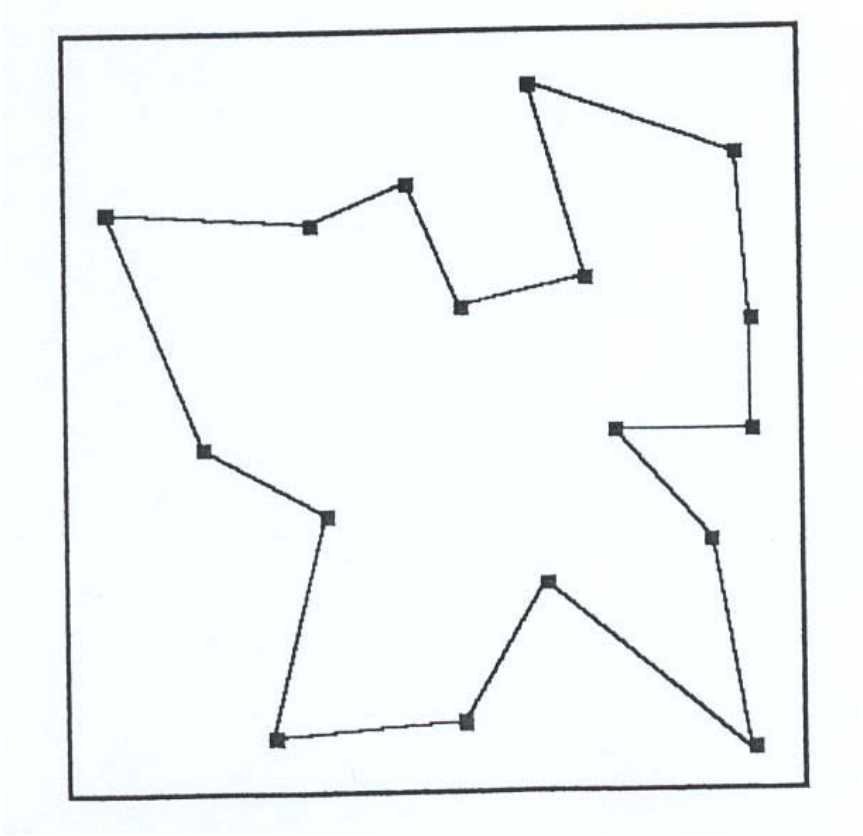
Population: 61 chromosomes - Elitism

Crossover: OX (P_c = 0,6)

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

TSP

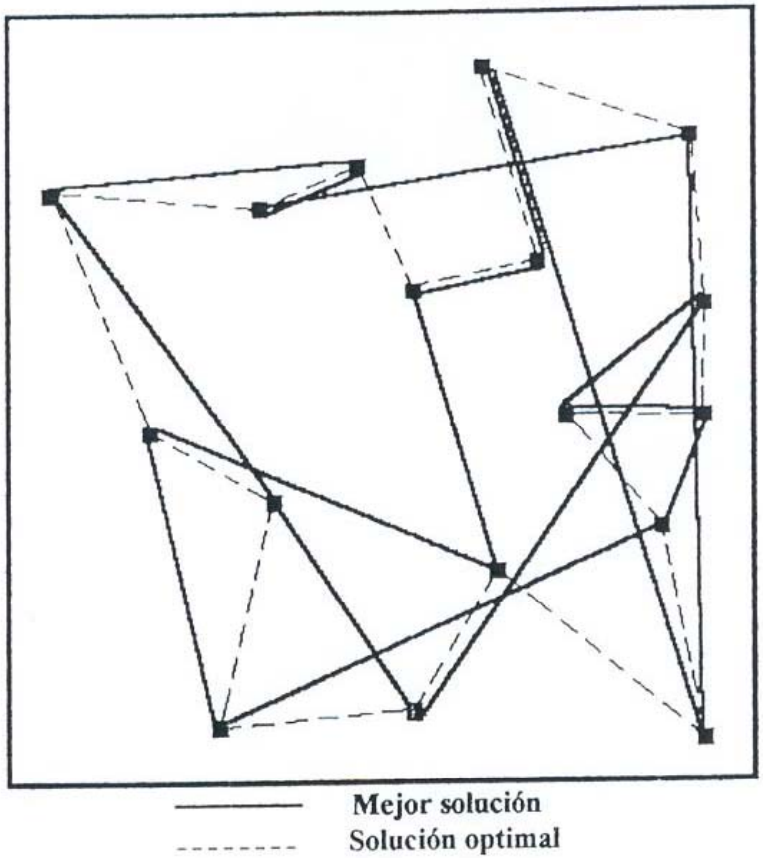


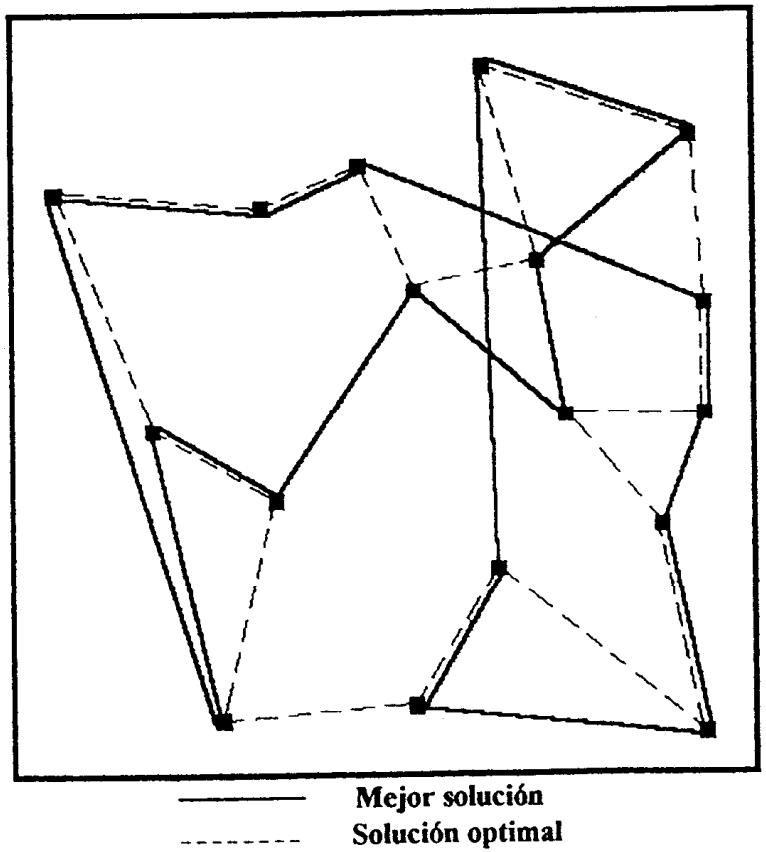


17! = 3.5568743 e14 possible solutions

Optimum solution: 226.64

TSP



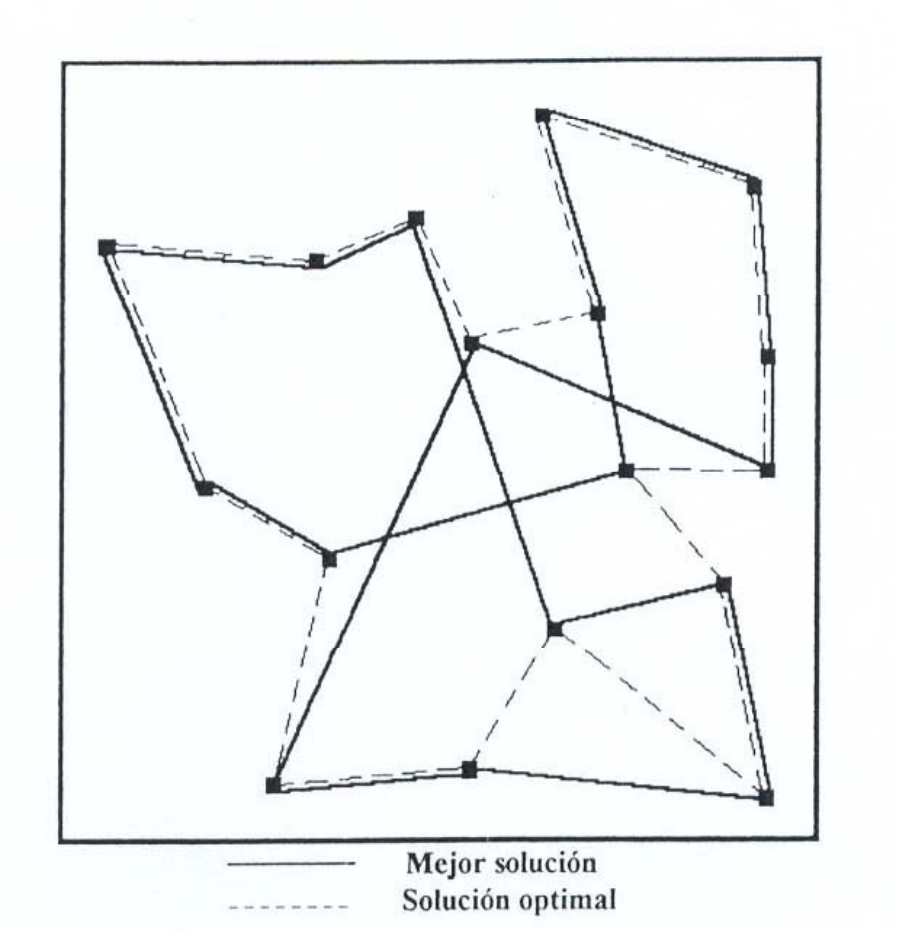


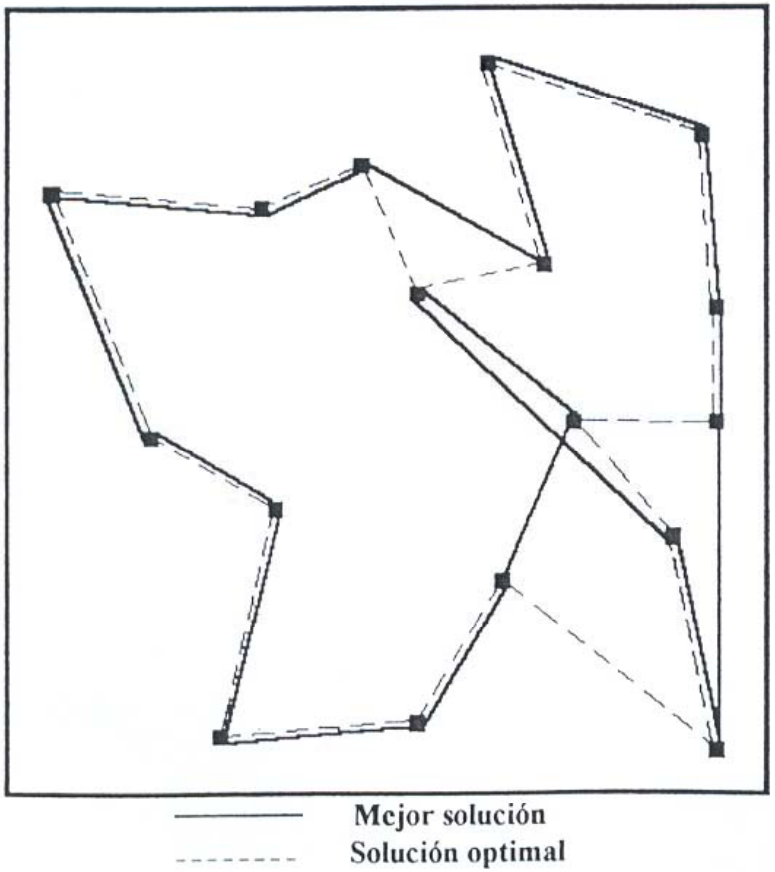
Iteration: 0 Cost: 403.7

Iteration: 25 Cost: 303.86

Optimum solution: 226.64

TSP



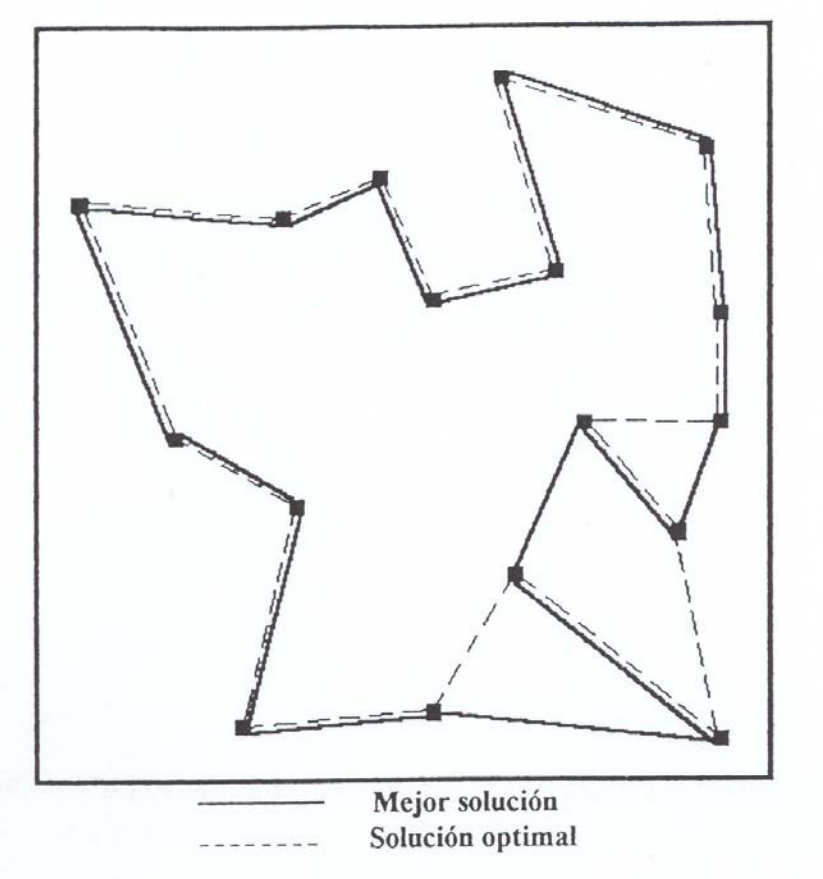


Iteration: 50 Cost: 293,6

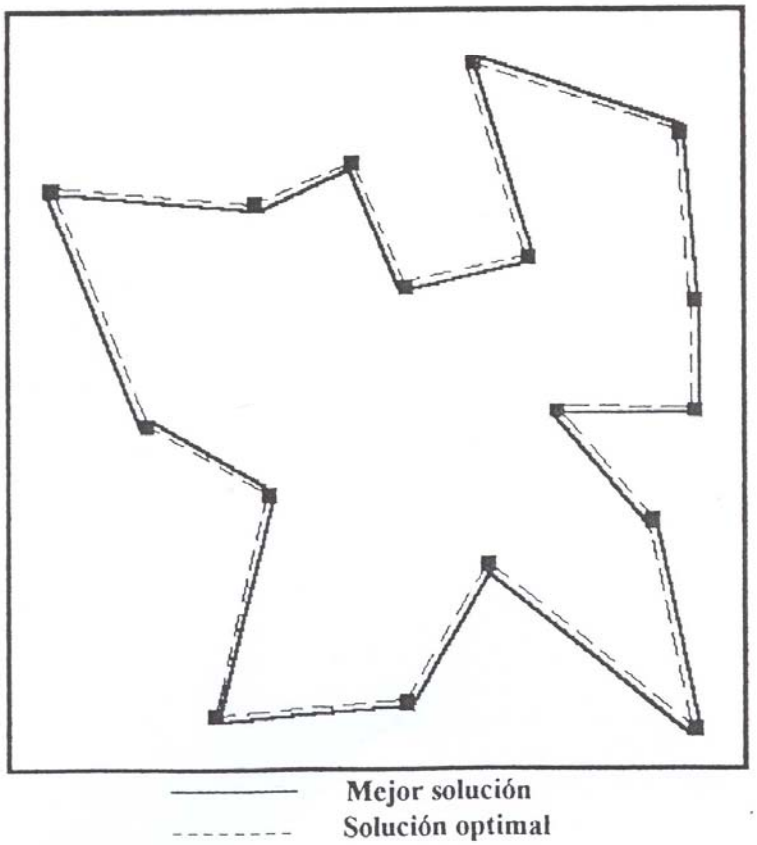
Iteration: 100 Cost: 256,55

Optimum solution: 226,64

TSP



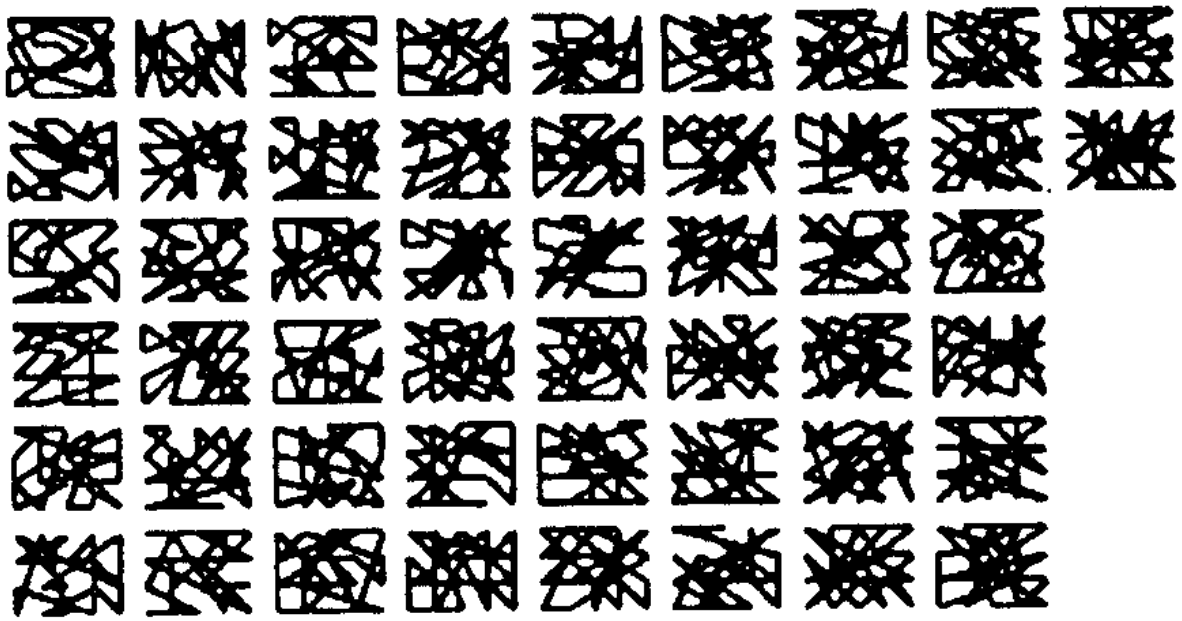
Iteration: 200 Costo: 231,4



Iteration: 250

Optimum solution: 226,64

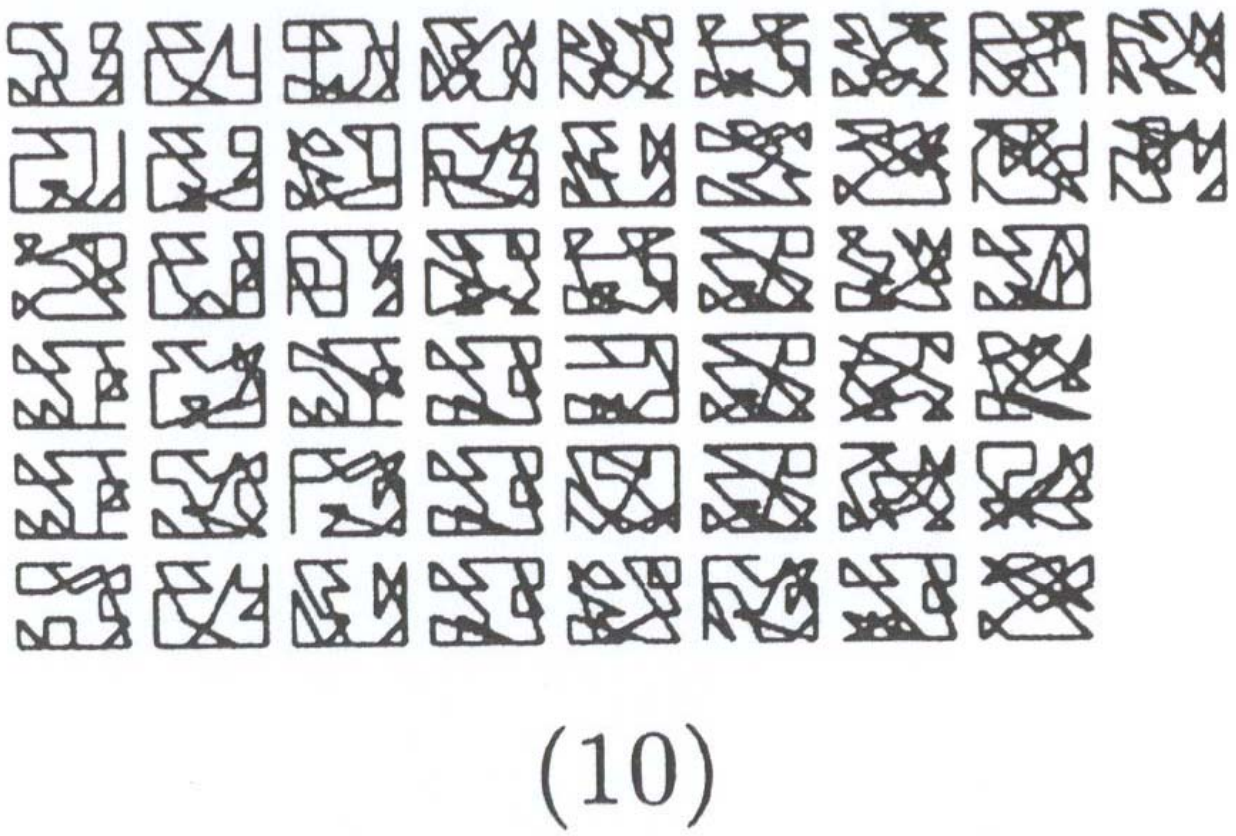
TSP



(0)

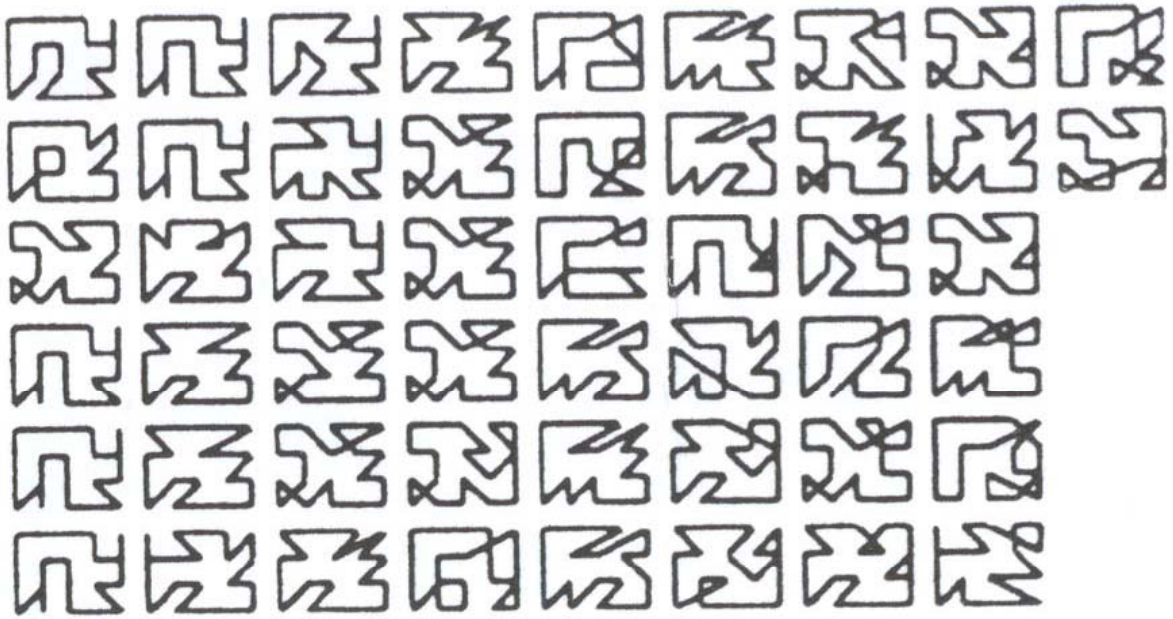
Visualization of the evolution with a population of size 50 and 70 iterations

TSP



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

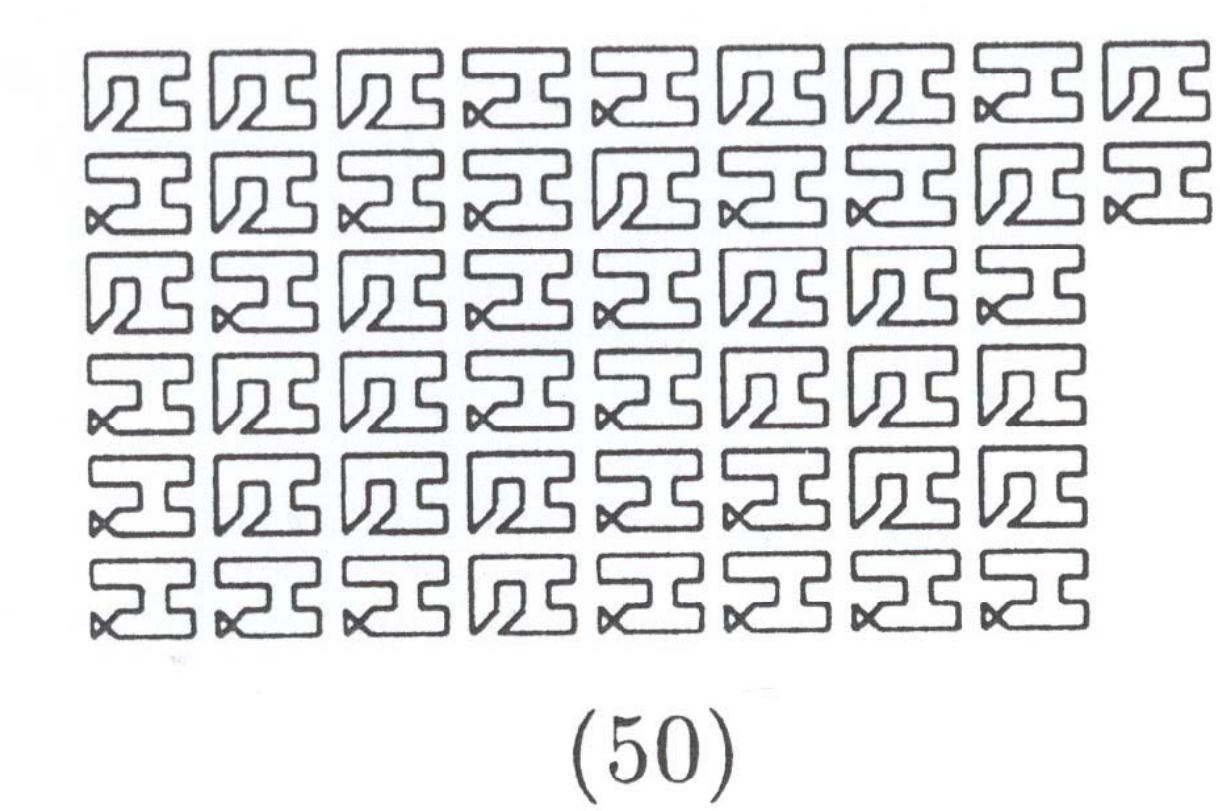
TSP



(30)

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

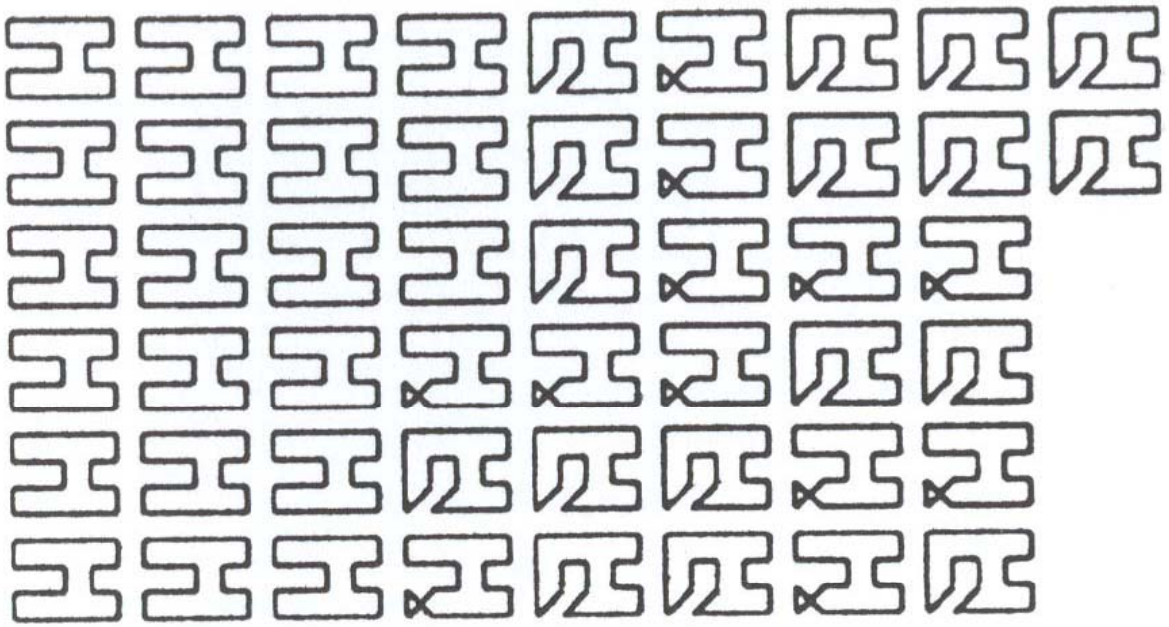
TSP



(50)

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

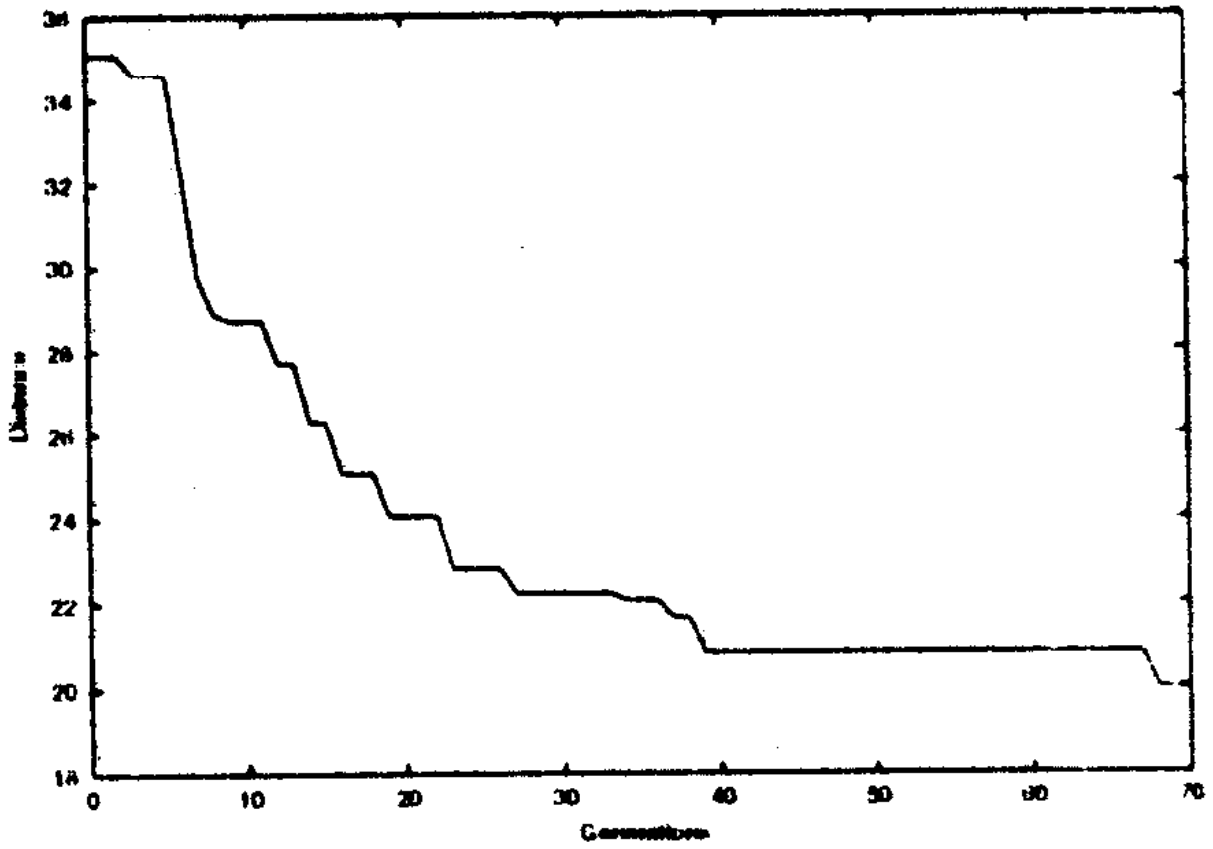
TSP



(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/

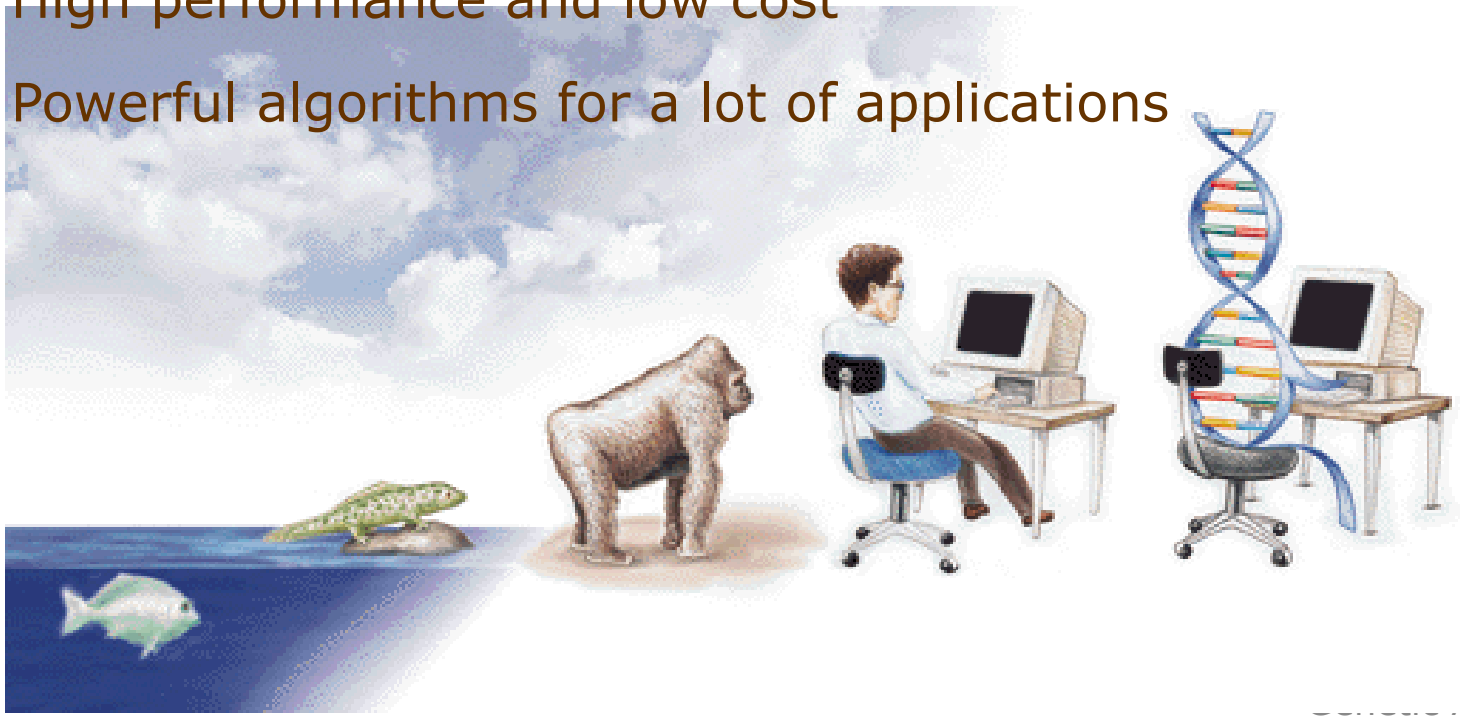
Maintained: Sebastián Ventura, Universad de Córdoba (sventura@uco.es)

S. Ventura, C. Romero, A. Zafra, J.A. Delgado, C. Hervás-Martínez. JCLEC: A Java Framework for Evolutionary Computing. Soft Computing, 2007, in press.

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 DISTRIBUTIION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIOARY 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)

Outline

- ✓ Introduction to Soft Computing/Computational Intelligence
- ✓ Fuzzy Sets and Fuzzy Systems
- ✓ Evolutionary Computation
- ✓ Neural Netwoks
- ✓ Concluding Remarks

Neural Networks

- Typical NN structure for classification:
  - One output node per class
  - Output value is class membership function value
- Supervised learning
- For each tuple in training set, propagate it through NN. Adjust weights on edges to improve future classification.
- Algorithms: Propagation, Backpropagation, Gradient Descent

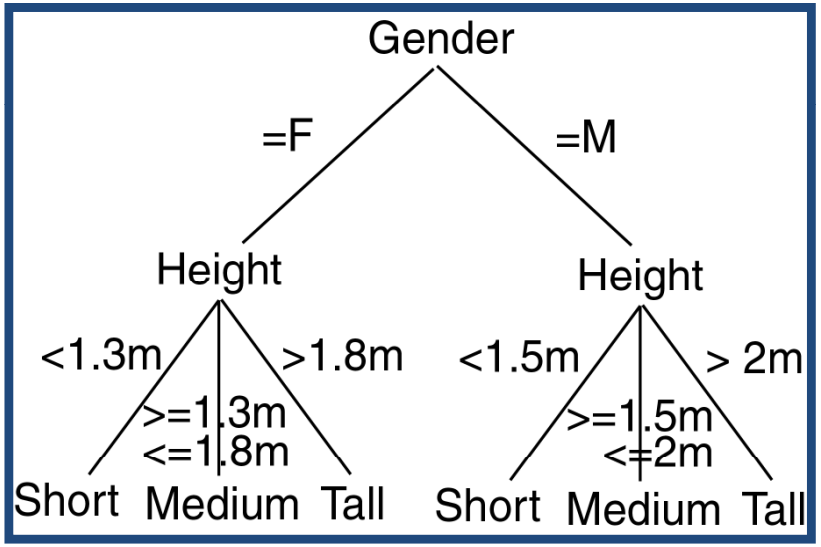
NN Issues

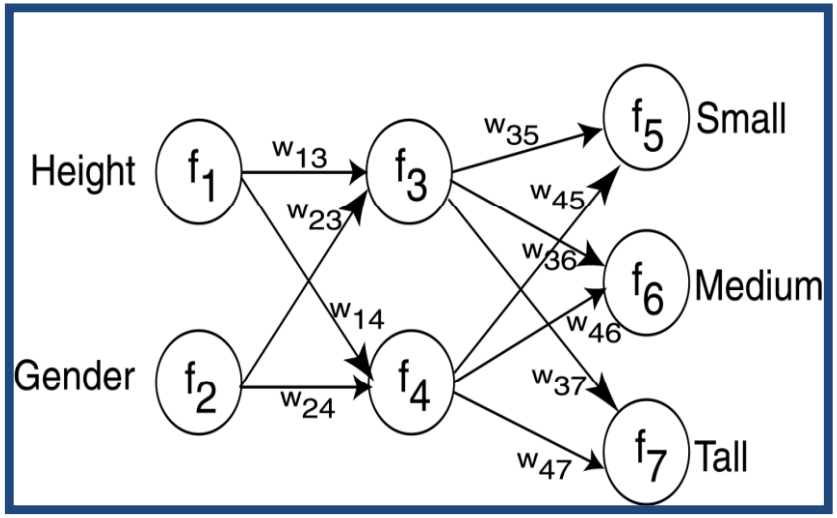
- Number of source nodes
- Number of hidden layers
- Training data
- Number of sinks
- Interconnections
- Weights
- Activation Functions
- Learning Technique
- When to stop learning

Classification with Neural Networks

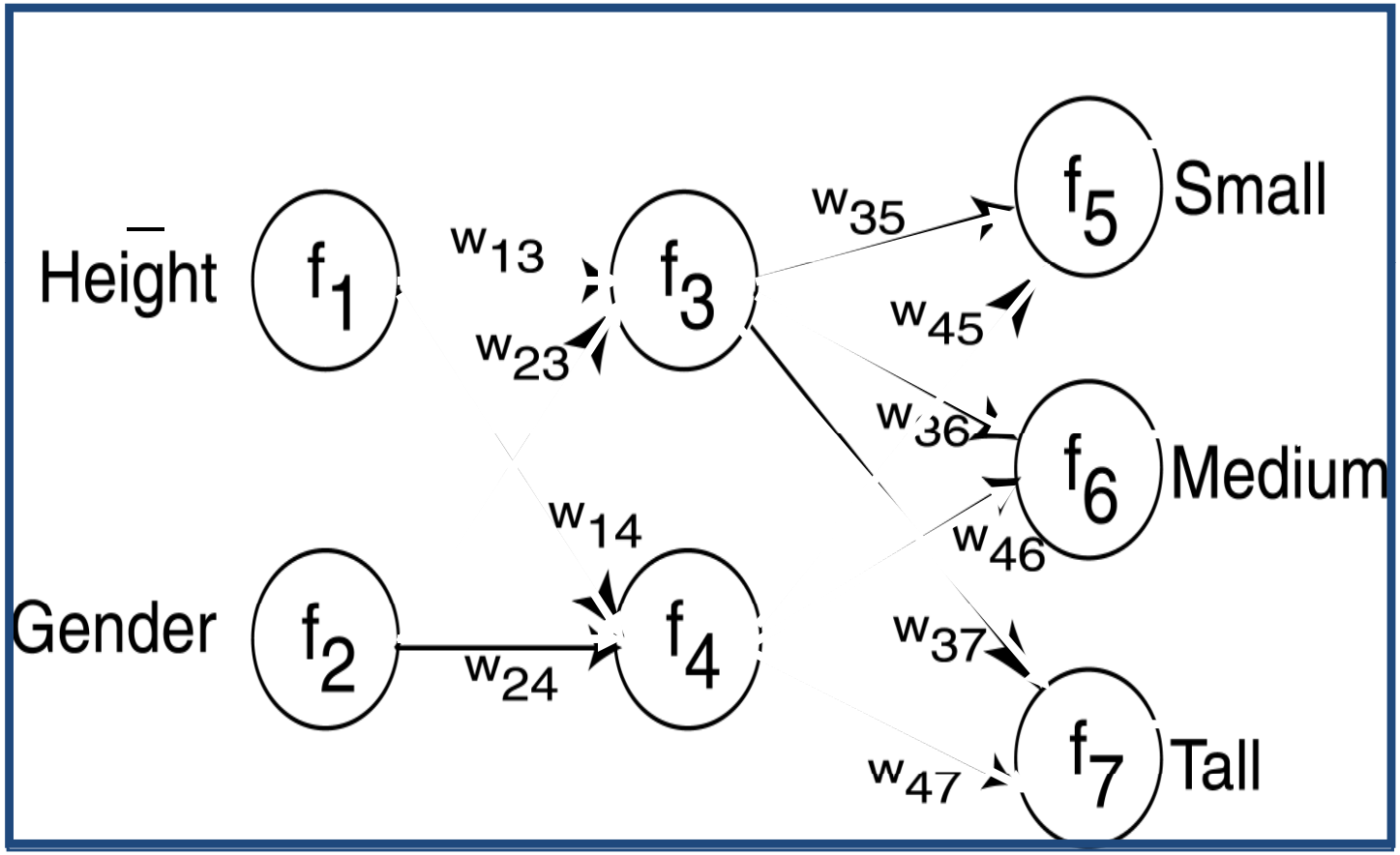
Decision Tree vs. Neural Network

From Rules to Black model





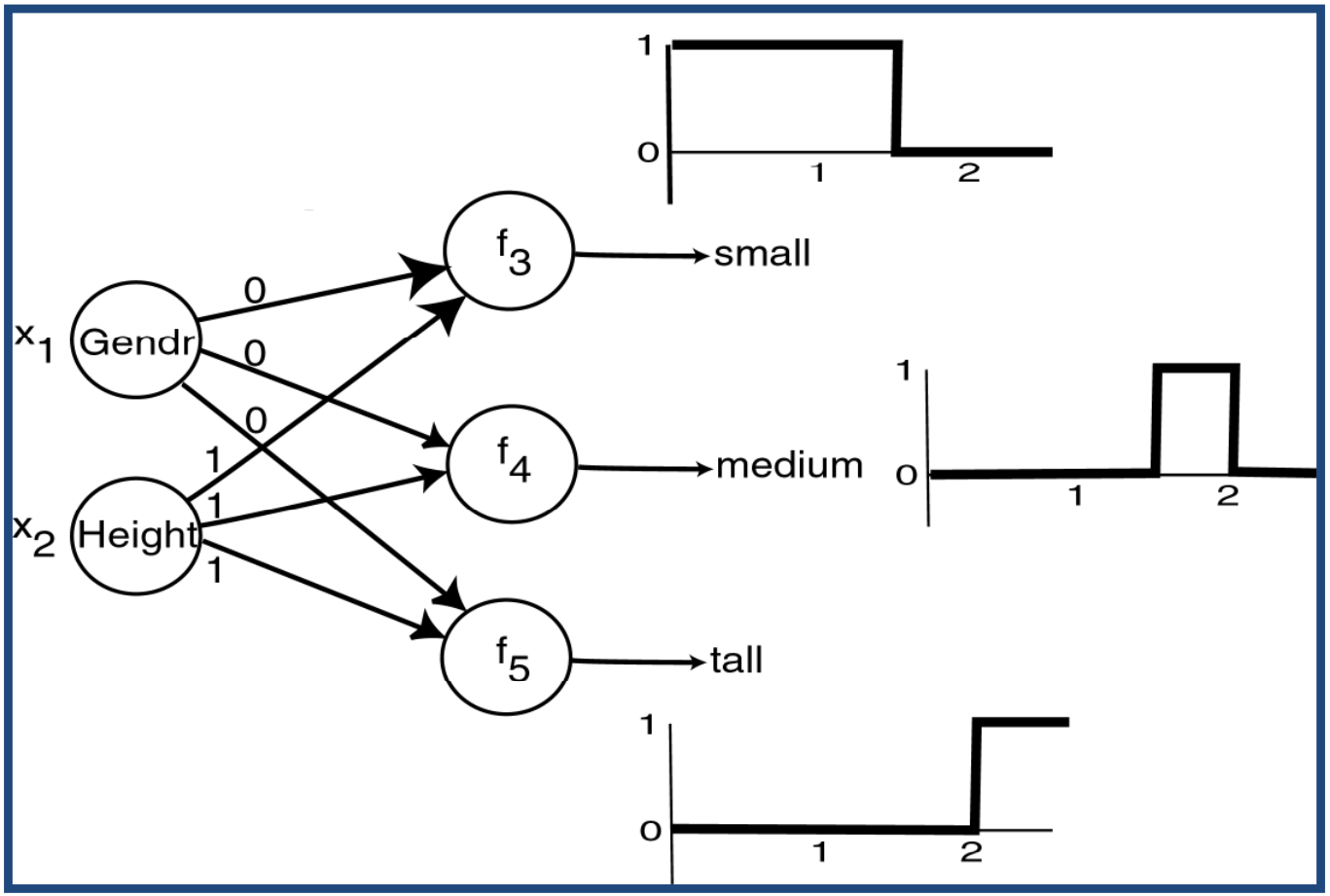
Propagation



NN Propagation Algorithm

Input:
N //Neural Network
$X =< x_1, ..., x_h >$ //Input tuple consisting of values for input attributes only
Output:
$Y =< y_1, ..., y_m >$ //Tuple consisting of output values from NN
Propagation Algorithm:
//Algorithm illustrates propagation of a tuple through a NN
for each node i in the input layer do
Output x_i on each output arc from i;
for each hidden layer do
for each node i do
$S_i = (\sum_{j=1}^{k}(w_{ji}\, x_{ji}))$;
for each output arc from i do
Output $\frac{(1-e^{-S_i})}{(1+e^{-c\, S_i})}$;
for each node i in the output layer do
$S_i = (\sum_{j=1}^{k}(w_{ji}\, x_{ji}))$;
Output $y_i = \frac{1}{(1+e^{-c\, S_i})}$;

Example Propagation



NN Learning

- Adjust weights to perform better with the associated test data.
- ***Supervised:*** Use feedback from knowledge of correct classification.
- ***Unsupervised:*** No knowledge of correct classification needed.

NN Supervised Learning

```
Input:
    N      //Starting Neural Network
    X      //Input tuple from Training Set
    D      //Output tuple desired
Output:
    N      //Improved Neural Network
SupLearn Algorithm:
                //Simplistic algorithm to illustrate approach to NN learning
    Propagate X through N producing output Y;
    Calculate error by comparing D to Y;
    Update weights on arcs in N to reduce error;
```

Supervised Learning

- Possible error values assuming output from node i is y_i but should be d_i:

$$| y_i - d_i |$$

$$\frac{(y_i - d_i)^2}{2}$$

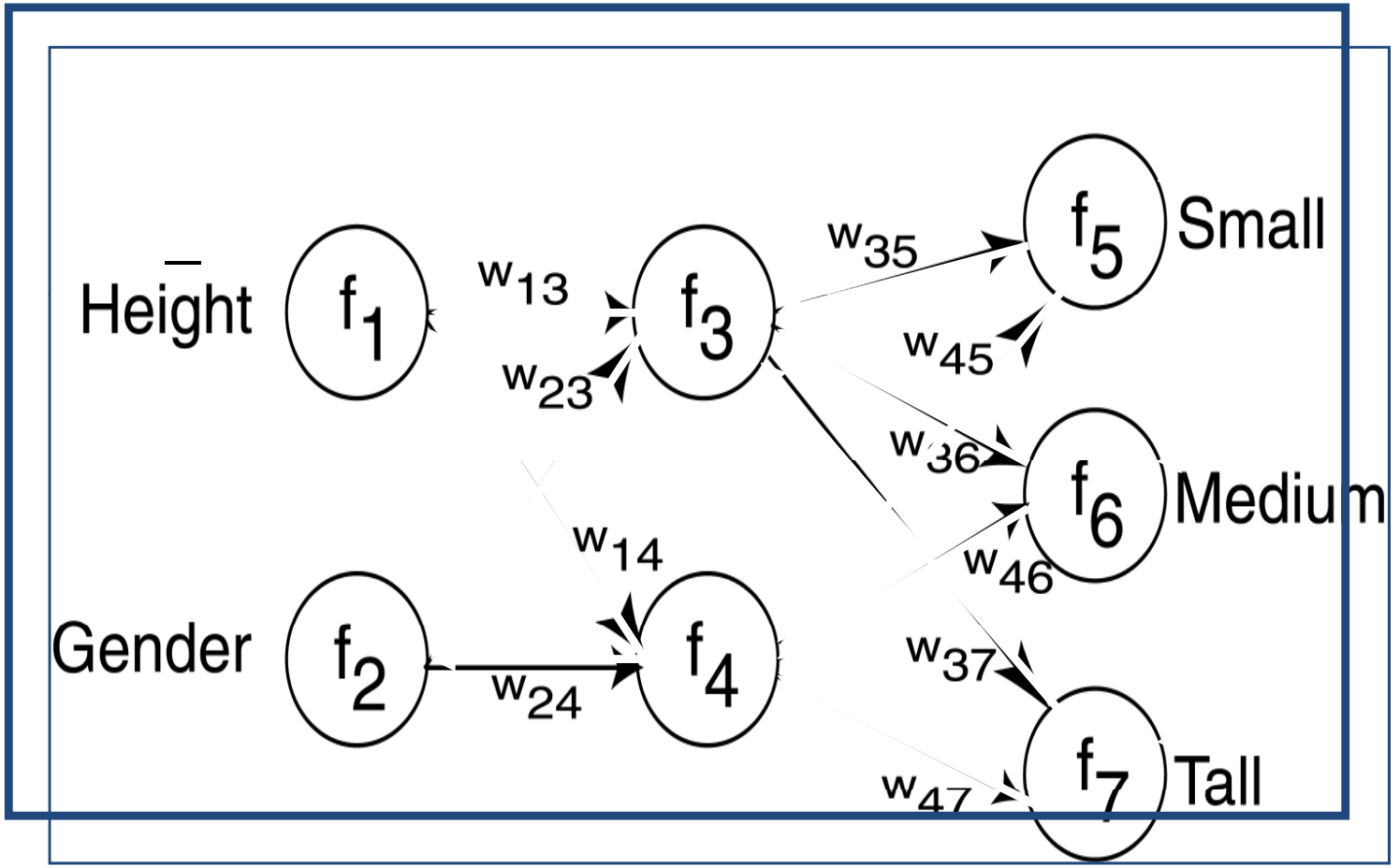
$$\sum_{i=1}^{m} \frac{(y_i - d_i)^2}{m}$$

- Change weights on arcs based on estimated error

NN Backpropagation

- Propagate changes to weights backward from output layer to input layer.
- ***Delta Rule:*** $\triangle w_{ij} = c\, x_{ij} (d_j - y_j)$
- ***Gradient Descent:*** technique to modify the weights in the graph.

Backpropagation



Backpropagation Algorithm

Input:

N //Starting Neural Network

$X =< x_1, ..., x_h >$ //Input tuple from Training Set

$D =< d_1, ..., d_m >$ //Output tuple desired

Output:

N //Improved Neural Network

Backpropagation Algorithm:

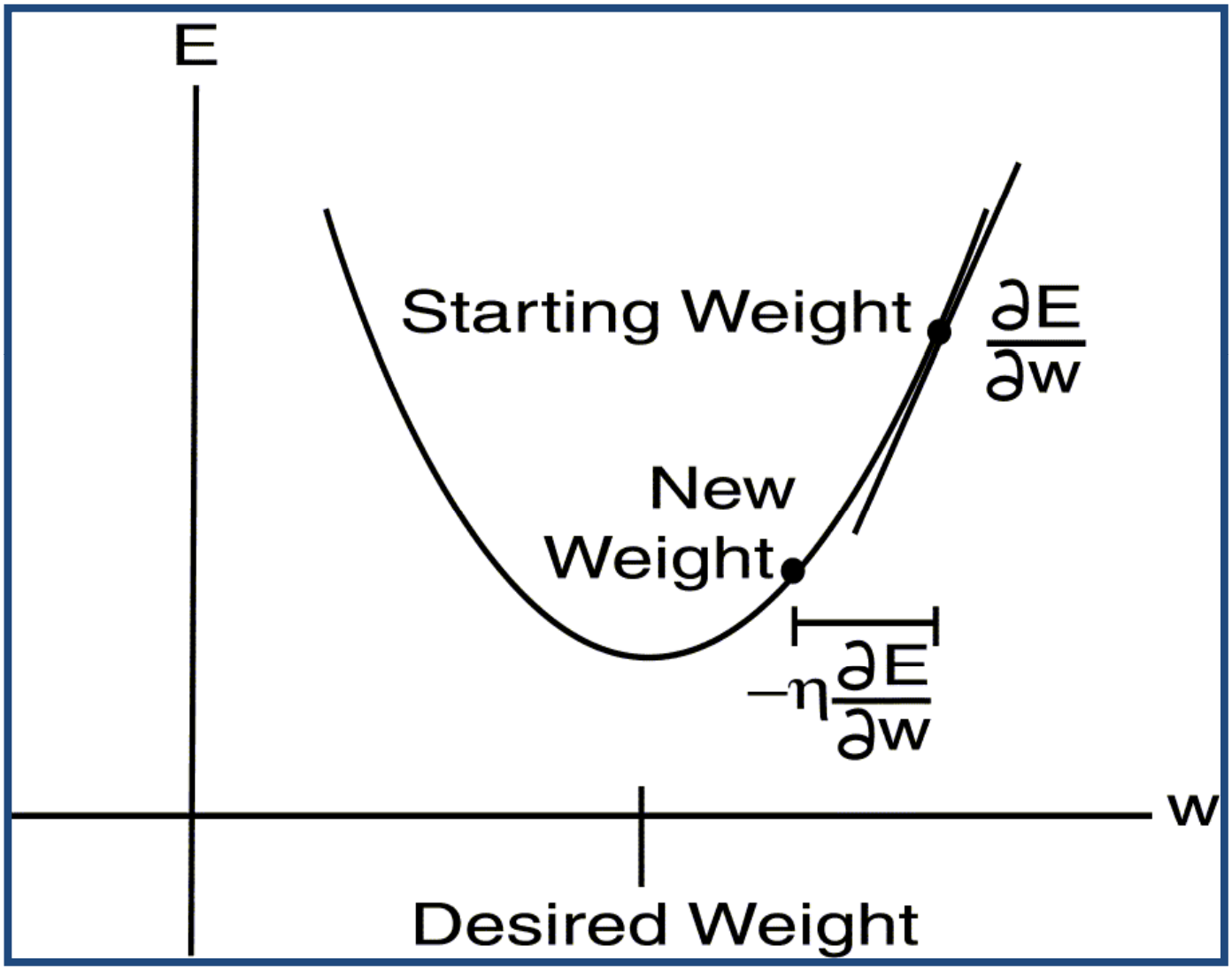
//Illustrate backpropagation

Propagation(N, X);

$E = 1/2 \sum_{i=1}^{m} (d_i - y_i)^2$;

Gradient(N,E);

Gradient Descent



Gradient Descent Algorithm

Input:
N //Starting Neural Network
E //Error found from Back algorithm
Output:
N //Improved Neural Network
Gradient Algorithm:
//Illustrates incremental gradient descent
for each node i **in output layer do**
for each node j **input to** i **do**
$\Delta w_{ji} = \eta\ (d_i - y_i)\ y_j\ (1 - y_i)\ y_i$;
$w_{ji} = w_{ji} + \Delta w_{ji}$;
layer = previous layer;
for each node j **in this layer do**
for each node k **input to** j **do**
$\Delta w_{kj} = \eta\ y_k\ \frac{1-(y_j)^2}{2} \sum_m (d_m - y_m)\ w_{jm}\ y_m(1 - y_m)$;
$w_{kj} = w_{kj} + \Delta w_{kj}$;

Output Layer Learning

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial S_i} \frac{\partial S_i}{\partial w_{ji}}$$

$$\frac{\partial y_i}{\partial S_i} = \frac{\partial}{\partial S_i}\left(\frac{1}{(1+e^{-S_i})}\right) = \left(1 - \left(\frac{1}{1+e^{-S_i}}\right)\right)\left(\frac{1}{1+e^{-S_i}}\right) = (1-y_i)y_i$$

$$\frac{\partial S_i}{\partial w_{ji}} = y_j.$$

$$\frac{\partial E}{\partial y_i} = \frac{\partial}{\partial y_i}(1/2 \sum_m (d_m - y_m)^2) = -(d_i - y_i)$$

$$\Delta w_{ji} = \eta\,(d_i - y_i)\,y_j\,\left(1 - \frac{1}{1+e^{-S_i}}\right)\frac{1}{1+e^{-S_i}} = \eta\,(d_i - y_i)\,y_j\,(1-y_i)\,y_i$$

Hidden Layer Learning

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}}$$

$$\frac{\partial E}{\partial w_{kj}} = \sum_m \frac{\partial E}{\partial y_m} \frac{\partial y_m}{\partial S_m} \frac{\partial S_m}{\partial y_j} \frac{\partial y_j}{\partial S_j} \frac{\partial S_j}{\partial w_{kj}}.$$

$$\frac{\partial E}{\partial y_m} = -(d_m - y_m)$$

$$\frac{\partial y_m}{\partial S_m} = (1 - y_m) y_m$$

$$\frac{\partial S_m}{\partial y_j} = w_{jm}$$

$$\frac{\partial y_j}{\partial S_j} = \frac{\partial}{\partial S_j}\left(\frac{(1 - e^{-S_j})}{(1 + e^{-S_j})}\right) = \frac{(1 + (\frac{1-e^{-S_j}}{1+e^{-S_j}}))\ (1 - (\frac{1-e^{-S_j}}{1+e^{-S_j}}))}{2} = \frac{1 - y_j^2}{2}$$

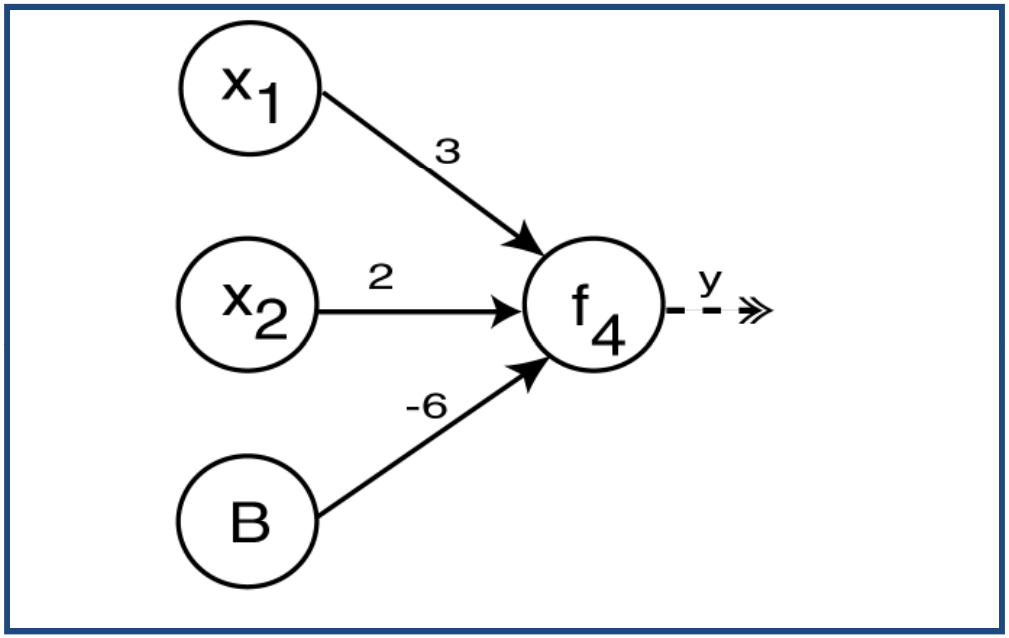
$$\frac{\partial S_j}{\partial w_{kj}} = y_k$$

$$\Delta w_{kj} = \eta\ y_k\ \frac{1 - (y_j)^2}{2} \sum_m (d_m - y_m)\ w_{jm}\ y_m\ (1 - y_m)$$

Types of NNs

- Different NN structures used for different problems.
- Perceptron
- Self Organizing Feature Map
- Radial Basis Function Network

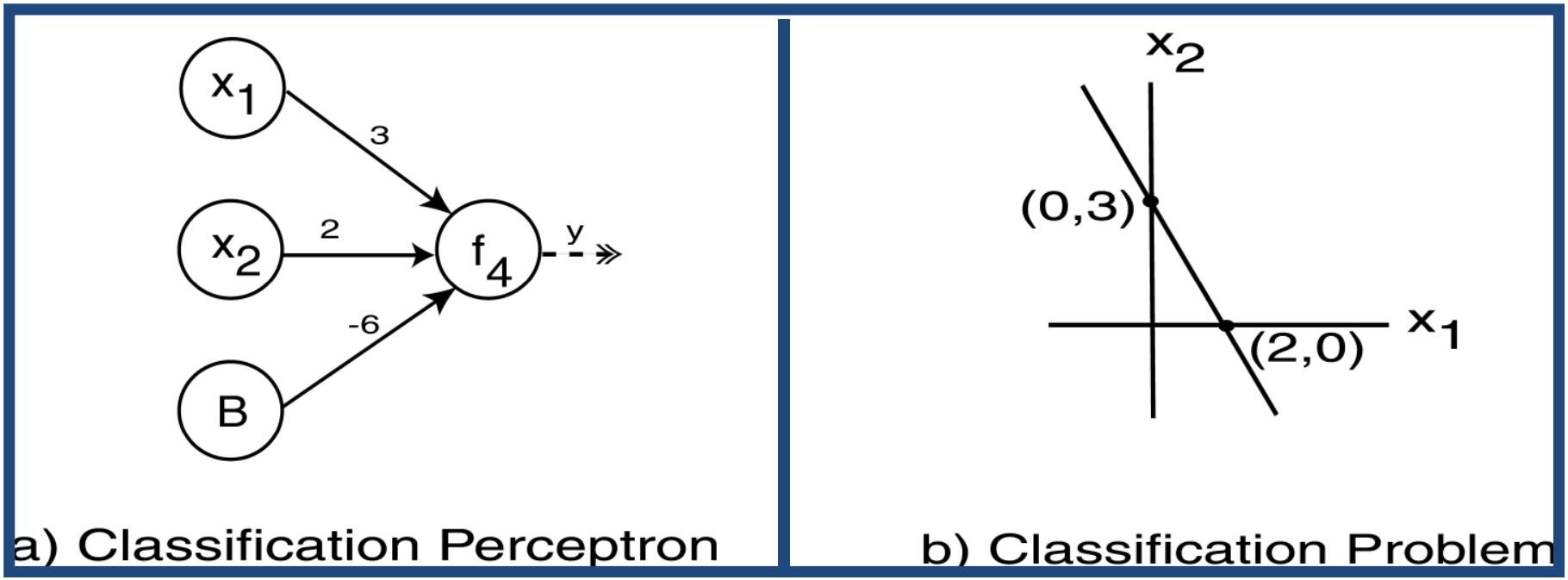
Perceptron



- Perceptron is one of the simplest NNs.
- No hidden layers.

Perceptron Example

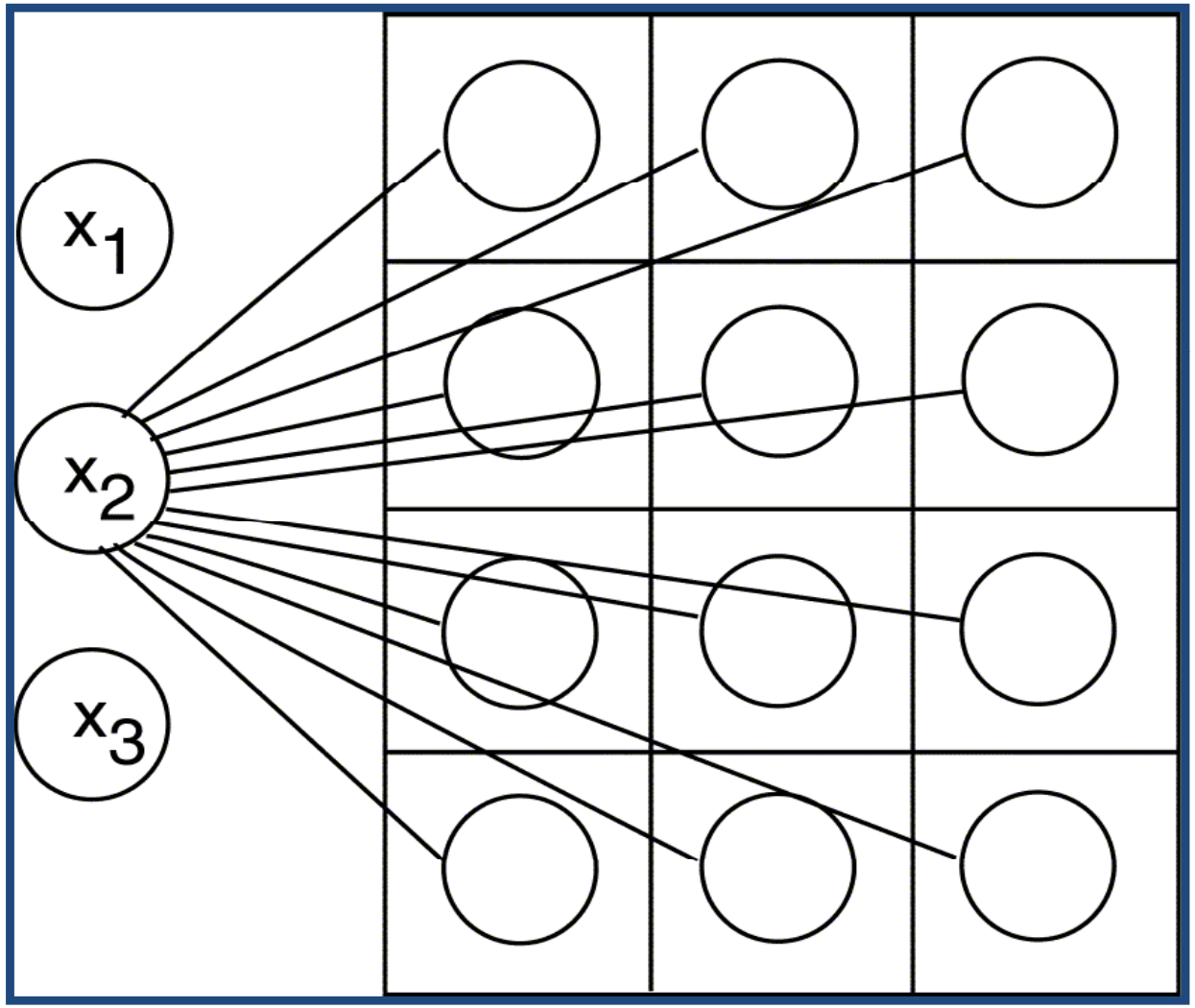
- Suppose:
  - Summation: $S=3x_1+2x_2-6$
  - Activation: if S>0 then 1 else 0



Self Organizing Feature Map (SOFM)

- Competitive Unsupervised Learning
- Observe how neurons work in brain:
  - Firing impacts firing of those near
  - Neurons far apart inhibit each other
  - Neurons have specific nonoverlapping tasks
- Ex: Kohonen Network

Kohonen Network



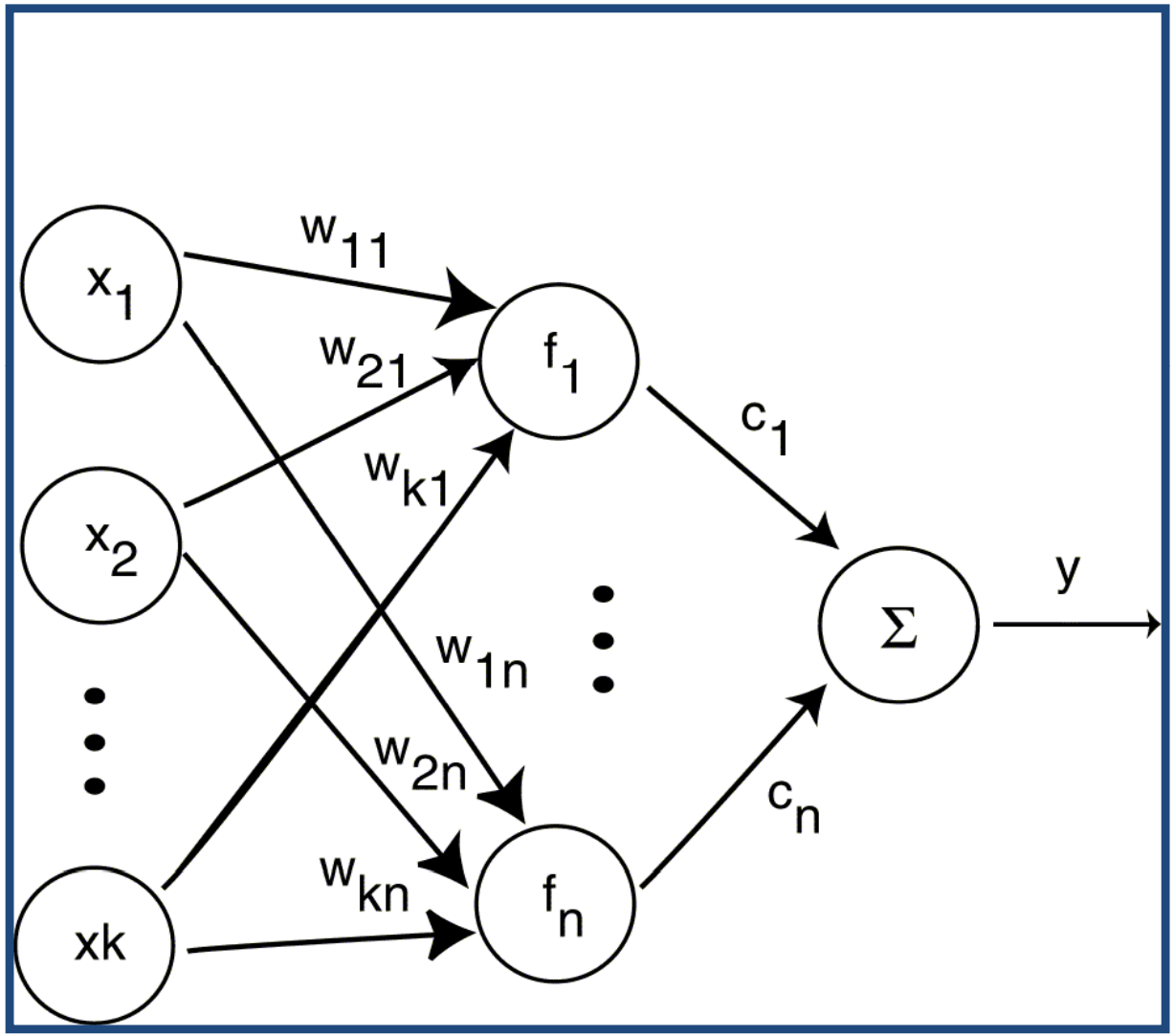
Kohonen Network

- Competitive Layer – viewed as 2D grid
- Similarity between competitive nodes and input nodes:
  - Input: $X = \langle x_1, \ldots, x_h \rangle$
  - Weights: $\langle w_{1i}, \ldots, w_{hi} \rangle$
  - Similarity defined based on dot product
- Competitive node most similar to input "wins"
- Winning node weights (as well as surrounding node weights) increased.

Radial Basis Function Network

- RBF function has Gaussian shape
- RBF Networks
  - Three Layers
  - Hidden layer – Gaussian activation function
  - Output layer – Linear activation function

Radial Basis Function Network



Outline

- ✓ Introduction to Soft Computing/Computational Intelligence
- ✓ Fuzzy Sets and Fuzzy Systems
- ✓ Evolutionary Computation
- ✓ Neural Netwoks
- ✓ Concluding Remarks

Concluding Remarks

Summary

- In many ways, soft computing represents a significant paradigm shift in the aims of computing - a shift which reflects the fact that the human mind, unlike present day computers, possesses a remarkable ability to store and process information which is pervasively imprecise, uncertain and lacking in categoricity.

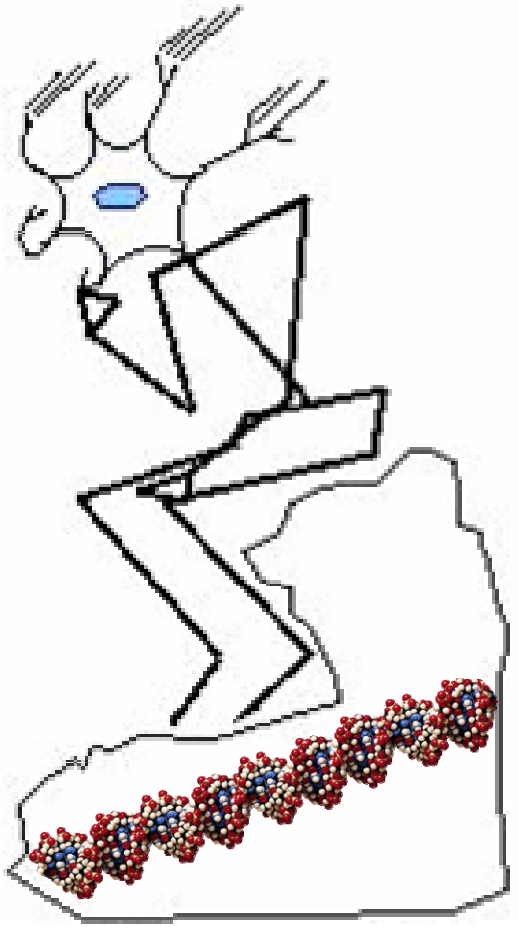
The hybridization of basic Computational Intelligence techniques is fundamental in Soft Computing.

Concluding Remarks

More on Hybrid models

In late eighties scientists thought –

Why NOT Integrations ?

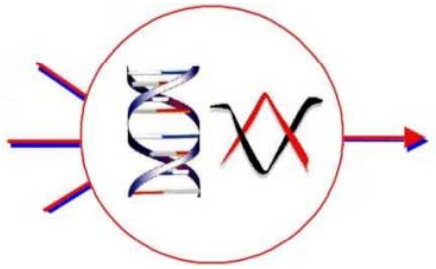


Neuro fuzzy models
Genetic fuzzy systems,
Evolutionary neural networks,
....

Concluding Remarks

In late eighties scientists thought –

Why NOT Integrations ?



Fuzzy Logic + ANN

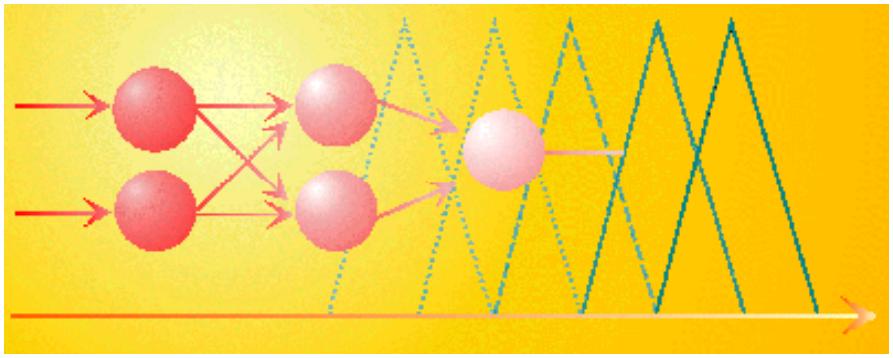
ANN + GA

Fuzzy Logic + ANN + GA

Fuzzy Logic + ANN + GA + Rough Set

Neuro-fuzzy hybridization is the most visible integration realized so far.

Concluding Remarks



Neuro fuzzy systems

A neuro-fuzzy system is a fuzzy system that uses a learning algorithm derived from or inspired by neural network theory to determine its parameters (fuzzy sets and fuzzy rules) by processing data samples.

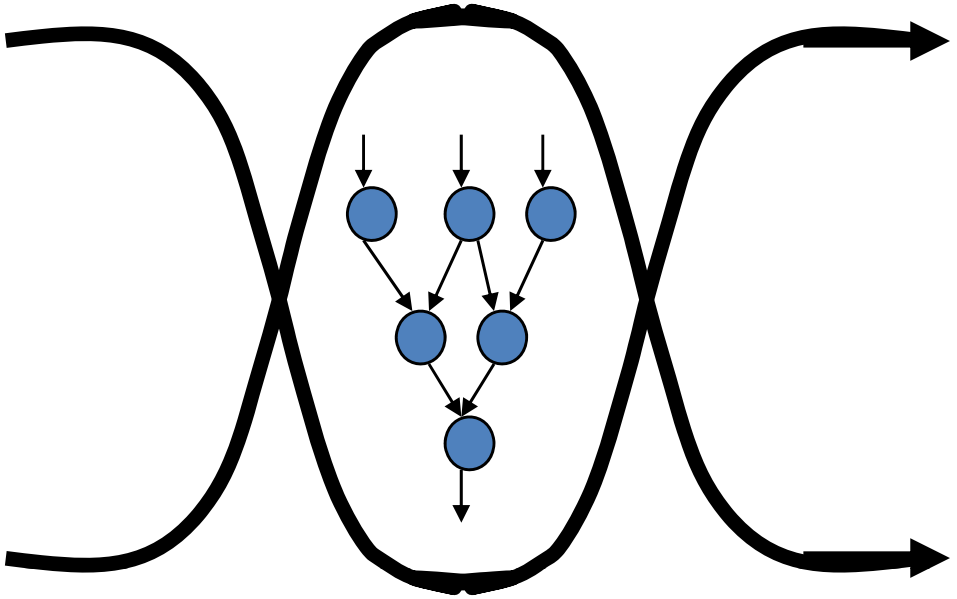
D. Nauck, F. Klawonn, and R. Kruse, Foundations of Neuro--Fuzzy Systems, (Wiley, Chichester, 1997)

Concluding Remarks

Fuzzy Set theoretic models try to mimic human reasoning and the capability of handling uncertainty – (SW)

Neural Network models attempt to emulate architecture and information representation scheme of human brain – (HW)

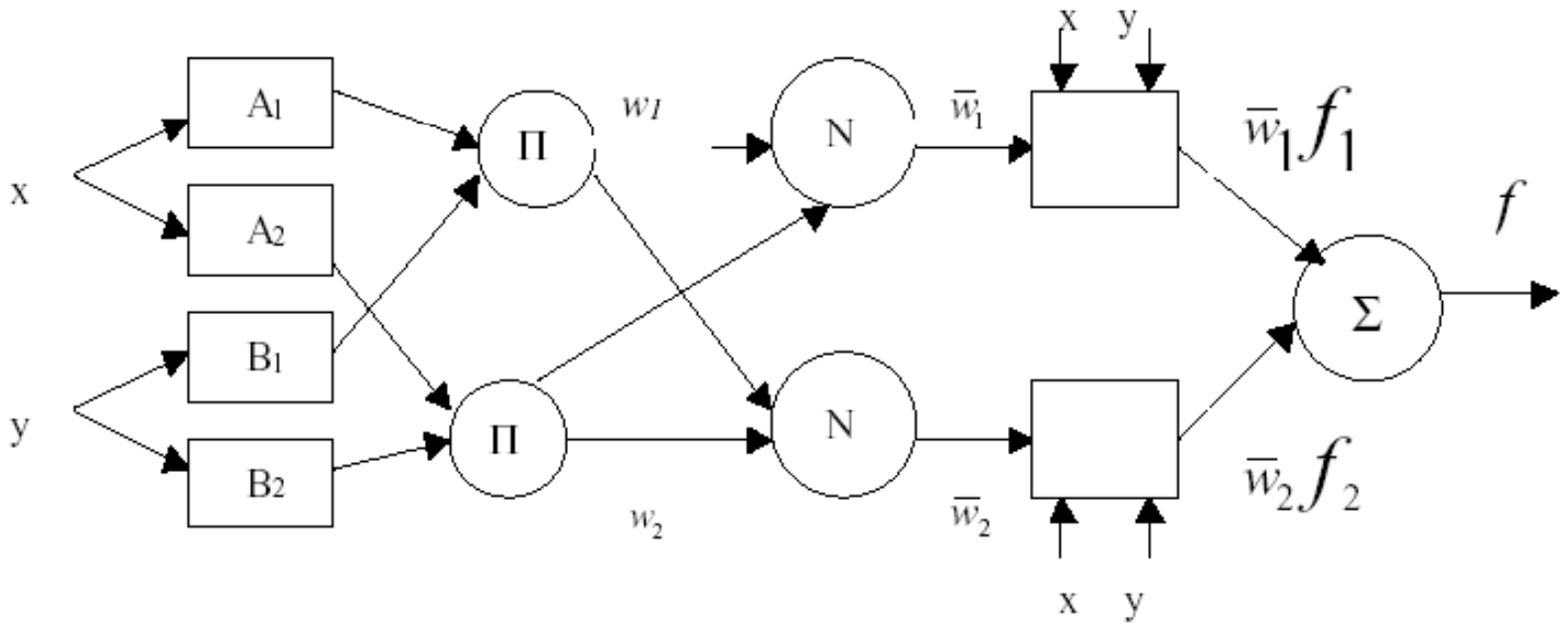
Why Fusion?



NEURO-FUZZY Computing

Concluding Remarks

Well known Neuro fuzzy systems

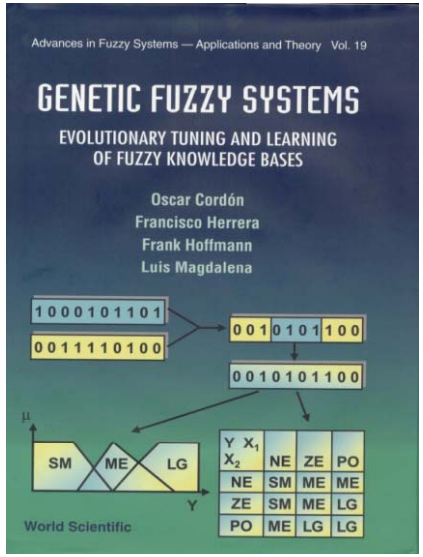


ANFIS: *Adaptive Network based Fuzzy Inference System*
(Jyh-Shing Roger Jang, 1993)

Concluding Remarks

Genetic fuzzy systems

A Genetic Fuzzy Systems is basically a fuzzy system augmented by a learning process based on a Genetic Algorithm.



GENETIC FUZZY SYSTEMS
Evolutionary Tuning and Learning of Fuzzy Knowledge Bases.
O. Cordón, F. Herrera, F. Hoffmann, L. Magdalena
World Scientific, July 2001.

Knowledge-based Systems

- Approximate reasoning
- Probabilistic reasoning
- Case based reasoning

Hybrid Systems

- Neuro-fuzzy
- Genetic neural
- Fuzzy genetic
- Fuzzy neuro genetic

Data Driven Systems

- Neural network system
- Evolutionary computing

Machine Intelligence

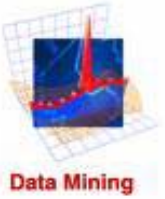
- Fuzzy logic
- Rough sets

Pattern recognition and learning

Non-linear Dynamics

- Chaos theory
- Rescaled range analysis (wavelet)
- Fractal analysis

Machine Intelligence: A core concept for grouping various advanced technologies with Pattern Recognition and Learning



Data Mining and Soft Computing

Summary

1. Introduction to Data Mining and Knowledge Discovery
2. Data Preparation
3. Introduction to Prediction, Classification, Clustering and Association
4. Data Mining - From the Top 10 Algorithms to the New Challenges
5. Introduction to Soft Computing. Focusing our attention in Fuzzy Logic and Evolutionary Computation
6. Soft Computing Techniques in Data Mining: Fuzzy Data Mining and Knowledge Extraction based on Evolutionary Learning
7. Genetic Fuzzy Systems: State of the Art and New Trends
8. Some Advanced Topics I: Classification with Imbalanced Data Sets
9. Some Advanced Topics II: Subgroup Discovery
10. Some advanced Topics III: Data Complexity
11. Final talk: How must I Do my Experimental Study? Design of Experiments in Data Mining/Computational Intelligence. Using Non-parametric Tests. Some Cases of Study.