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.
What is a genetic algorithm?

Artificial Evolution

There are 4 classic paradigms:

Genetic Algorithms. 1975, Michigan University

Evolution Strategies 1964, Technische Universität Berlin

Evolutionary Programming. 1960-1966, Florida

Genetic Programming. 1989, Stanford University

There exist other models based on population evolution.
What is a genetic algorithm?

Artificial Evolution
The ingredients

- **t**
- **reproduction**
- **selection**
- **t + 1**
- **mutation**
- **Crossover (or recombination)**
The evolution cycle

Selection -> PARENTS

Crossover
Mutation

Replacement -> POPULATION

DESCENDANTS

F. Herrera - Introduction to Genetic Algorithms
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 \( P(t-1) \)

\( P''(t) \) ← 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
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

CHROMOSOME

1 0 1 0 0 0 1 1

GEN
Example: Binary representation

8 bits genotype

1 0 1 0 0 0 1 1

Fenotype

- integer
- real number
- ...
- ¿Others?
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:

\[ f : \mathbb{R}^n \rightarrow \mathbb{R} \]
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 \( \ldots \) (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 (e.g., 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?

Selection

PARENTS

Representation
Initialization
Population
Fitness function

POPULATION
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.

F. Herrera - Introduction to Genetic Algorithms
HOW TO CONSTRUCT A GA?

Selection

PARENTS

Crossover

Representation
Initialization
Population
Fitness function

POPULATION
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): Biologica 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

```
Parent
0011100110100001111100

Offsprings
0101111000100001111001
0011100111001010101100

Parent
010111100101010101001
```
Example: uniform crossover

```
abcfgh
A BCDEF GH
```

```plaintext
abCdefgH
```
Example: Real coding crossover operator

Arithmetic crossover:

\[
\begin{array}{ccccccc}
  a & b & c & d & e & f \\
  A & B & C & D & E & F
\end{array}
\]

\[
\begin{array}{ccccccc}
  (a+A)/2 & (b+B)/2 & (c+C)/2 & (d+D)/2 & (e+E)/2 & (f+F)/2
\end{array}
\]
Example: real coding crossover operator $\text{BLX-} \alpha$

- Two chromosomes
  \[ C_1 = (c_{11}, \ldots, c_{1n}) \text{ } y \text{ } C_2 = (c_{21}, \ldots, c_{2n}) \, , \]

- $\text{BLX-} \alpha$ generates two descendants
  \[ H_k = (h_{k1}, \ldots, h_{ki}, \ldots, h_{kn}) \, , \text{ } k = 1, 2 \]

- where $h_{ki}$ is a random value in the interval:
  \[ [C_{\text{min}} - I \cdot \alpha, C_{\text{max}} + I \cdot \alpha] \]

- $C_{\text{max}} = \max \{c_{1i}, c_{2i}\}$
- $C_{\text{min}} = \min \{c_{1i}, c_{2i}\}$
- $I = C_{\text{max}} - C_{\text{min}}$, $\alpha \in [0, 1]$
Ejemplo: Operador de cruce para representación real: **BLX-α**

Exploration

\[ c_{\text{min}} - \alpha \cdot I \]

\[ I \]

Exploration

\[ c_{\text{max}} + \alpha \cdot I \]

Exploitation

\[ a_i \]

\[ c^1_i \]

\[ c^2_i \]

\[ b_i \]
Example: Crossover operator for order representation: OX

Parent 1

7 3 1 8 2 4 6 5

Parent 2

4 3 2 8 6 7 1 5

7, 3, 4, 6, 5

Order

4, 3, 6, 7, 5

Hijo 1

7 5 1 8 2 4 3 6
HOW TO CONSTRUCT A GA?

Selection

PARENTS

Crossover
Mutation

DESCENDANTS

HOW TO CONSTRUCT A GA?

Selection

PARENTS

Crossover
Mutation

DESCENDANTS

HOW TO CONSTRUCT A GA?

Selection

PARENTS

Crossover
Mutation

DESCENDANTS

HOW TO CONSTRUCT A GA?

Selection

PARENTS

Crossover
Mutation

DESCENDANTS

HOW TO CONSTRUCT A GA?

Selection

PARENTS

Crossover
Mutation

DESCENDANTS

HOW TO CONSTRUCT A GA?

Selection

PARENTS

Crossover
Mutation

DESCENDANTS

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
  • $\sigma$ is the typical deviation

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

For each parameter.
Example: order representation mutation

\[
\begin{array}{ccccccc}
7 & 3 & 1 & 8 & 2 & 4 & 6 & 5 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
7 & 3 & 6 & 8 & 2 & 4 & 1 & 5 \\
\end{array}
\]
HOW TO CONSTRUCT A GA?

1. Selection
2. Crossover
3. Mutation
4. Replacement

POPULATION
- Representation
- Initialization
- Population
- Fitness function

PARENTS

DESCENDANTS
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

Mutation

Replacement

Representation

Initialization

Population

Fitness function

POPULATION

DESCENDANTS

F. Herrera - Introduction to Genetic Algorithms
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

\[ P_{\text{actual}}(t) \]
\[ C_1 \]
\[ C_2 \]
\[ \ldots \]
\[ C_M \]

\[ \text{SELECTION} \]
\[ P_{\text{parents}} \]
\[ C'_1 \]
\[ C'_2 \]
\[ \ldots \]
\[ C'_M \]

\[ \text{CROSSOVER WITH PROB. } P_c \]
\[ C''_1 \]
\[ C''_2 \]
\[ \ldots \]
\[ C''_M \]

\[ \text{MUTATION WITH PROB. } P_m \]

\[ H_1 \]
\[ H_2 \]
\[ \ldots \]
\[ H_{M-1} \]
\[ C_1 \]

\[ \text{REPLACEMENT} \]
With elitism (maintaining the best individual from \( P(t) \))

\[ P_{\text{offspring}} \]
\[ H_1 = C'''_{m_1} \]
\[ H_2 = C'''_{m_2} \]
\[ \ldots \]
\[ H_M = C'''_{m} \]

\[ t \leftarrow t + 1 \]
Steady state model

\[ P_{\text{actual}}(t) \]
\[ C_1 \]
\[ C_2 \]
\[ \ldots \]
\[ C_M \]

\[ \rightarrow \]  
\[ P_{\text{selection}} \]

\[ P_{\text{parents}} \]
\[ C'_1 \]
\[ C'_2 \]

\[ \rightarrow \]  
\[ P_{\text{crossover}} \]

\[ P_{\text{intermediate}} \]
\[ C''_1 \]
\[ C''_2 \]

\[ \downarrow \]  
\[ P_{\text{mutation}} \]

\[ \text{with prob.} P_m \]

\[ \rightarrow \]  
\[ P_{\text{offspring}} \]

\[ H_1 = C''_1 \]
\[ H_2 = C''_{m_2} \]

\[ t \leftarrow t+1 \]
4. APPLICATIONS

- Structural optimization
- Learning
- Circuits design VLSI
- Clasification
- Control
- Trayectory generation
- Planification
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

$17! = 3.5568743 \times 10^{14}$ possible solutions

Optimum solution: 226.64

F. Herrera - Introduction to Genetic Algorithms
TSP

Iteration: 0  Cost: 403.7

Iteration: 25  Cost: 303.86

Optimum solution: 226.64

F. Herrera - Introduction to Genetic Algorithms
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

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

F. Herrera - Introduction to Genetic Algorithms
TSP

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

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

F. Herrera - Introduction to Genetic Algorithms
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

F. Herrera - Introduction to Genetic Algorithms
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)

6. SOFTWARE AND IMPLEMENTATIONS

JCLEC Features

- **Wide audience.** Software can be used by both expert people (EC researchers) and novice people (students or people from another field).
- **Several execution modes**
  - Batch mode (expert) based on a configuration file
  - Interactive mode (novice) based on a GUI with charting
- **Multiple individual encodings**
  - Linear genotype (binary, integer and real)
  - Tree genotype (expression tree and syntax tree)
  - Multiple crossover and mutation operators for each type of individual.
- **A variety of Evolutionary Algorithms**
  - Classical: SGA, Steady state and CHC
  - Multiobjective: SPEA2 and NSGA-II
  - Niching: Clearing, sequential and fitness sharing
  - Memetic: Generational and steady state
6. SOFTWARE AND IMPLEMENTATIONS

JCLEC Features (II)

• Extensible and reusable:
  • New problems can be easily added to the system.
  • New components (encodings, genetic operators and/or algorithms) can be easily added to the system.
• Multithreading
  • Speeds up evaluation in multiprocessor architectures
  • Parallel genetic algorithms implementation.
• Future work
  • New toolboxes planned:
    • Real optimization: ES, DE, MGG and GGG.
    • Evolutionary neural networks.
  • Native JCLEC. In complex problems, would be desirable to sacrifice the portability of the Java version to get a bigger efficiency (some applications increase their speed 10 times). This version has been developed using GNU gcj (java to native) compiler.
6. SOFTWARE AND IMPLEMENTATIONS

JCLEC Class Hierarchy

Evolutionary system and algorithm
contains

Data in an Evolutionary Algorithm
contains

Created
Evaluated
Evolutionary system and algorithm
contains

Individual commonalities

Operations performed in an EA

F. Herrera - Introduction to Genetic Algorithms
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

OTHER EVOLUTIONARY MODELS

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS

GENETIC ALGORITHMS

EVOLUTIONARY CLASSIC PARADIGMS

GENETIC PROGRAMMING

EVOLUTIONARY STRATEGIES

ESTIMATION DISTRIBUTION ALGORITHMS: PBIL, EDA, ...

PARTICLE SWARM: SOCIAL ADAPTATION

SCATTER SEARCH

CULTURAL EVOLUTIONARY ALGORITHMS

DIFFERENTIAL EVOLUTION

MEMETIC ALGORITHMS
BIBLIOGRAPHY

BOOKS - Classics

D.B. Fogel (Ed.) Evolutionary Computation. The Fossil Record.
   (Selected Readings on the History of Evolutionary Computation).

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)