
Evolutionary computation

Antoine CORNUÉJOLS - Christine MARTIN

AgroParisTech

Motivation

- **Mimicking natural evolution** to evolve better « solutions »
- Generation of successive **populations**
 - with **survival** and **reproduction** of the **fittests**
 - Using **mutation** and **cross-over** as reproduction operators
 - *Genotype* vs. *Phenotype*
- A kind of **generalized optimization method**
 - A **population** of “solutions” : *size*
 - Reproduction *operators*
 - *Selection* of the fittests

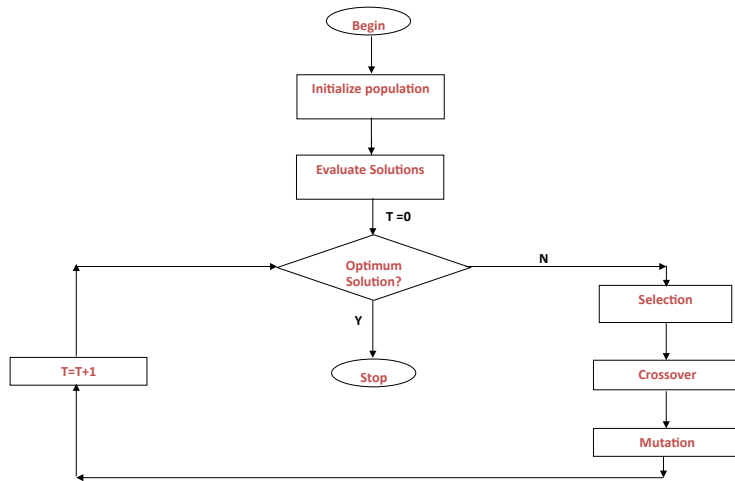
Outline

1. Motivation
2. Overview
3. Simple Genetic Algorithm (SGA)
4. Optimisation with GA
5. Theoretical foundations
6. Problem solving with GA : Genetic Programming
7. Co-evolution

Motivation : history

- “Evolutionary computing”
 - I. Rechenberg in the 60s.
 - *Optimization on real valued domains*
- Genetic algorithms
 - John Holland, “*Adaptation in Natural and Artificial Systems*”, 1975.
 - *Bit representation / Schema theorem / Problem-Solving method*
- Genetic Programming
 - John Koza, First book on Genetic Programming, 1992.
 - *Programs represented as trees*

Overview : principle



16/12/10

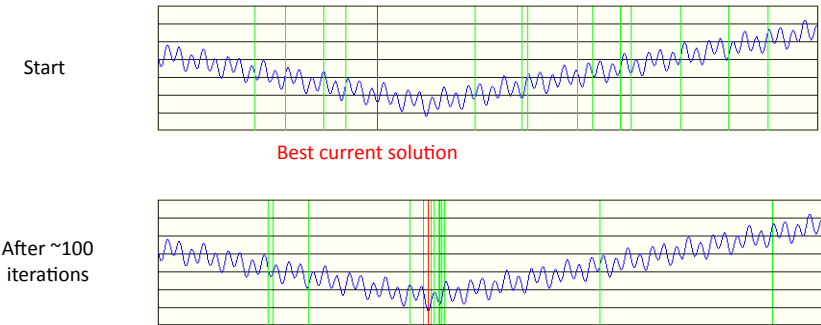
Evolutionary Computation - © A. Cornuéjols

5

Overview : illustration

Optimization of a function

http://cs.felk.cvut.cz/~xobitko/ga/example_f.html



16/12/10

Evolutionary Computation - © A. Cornuéjols

6

Simple GA

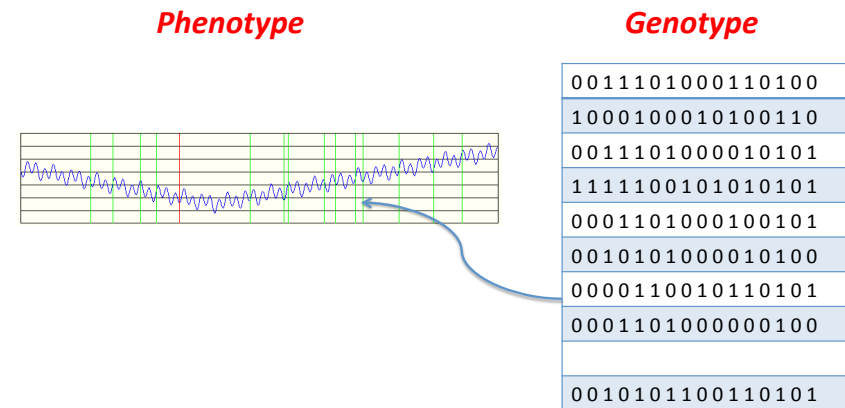
Natural Evolution	Evolutionary Computation
Population	Pool of solutions
Individual	Solution to a problem
Fitness of an individual	Quality of a solution
Chromosome	Encoding of a solution
Gene	Part of the encoding
Reproduction	Mutation and/or crossover

16/12/10

Evolutionary Computation - © A. Cornuéjols

7

Representation / chromosome encoding



16/12/10

Evolutionary Computation - © A. Cornuéjols

8

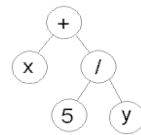
Representation / chromosome encoding

Various encoding schemes

- Bit strings
- Strings of values
- Real value
- tree

Chromosome 1	1 1 0 1 0 1 1 0 0 0 1
Chromosome 2	1 0 0 1 0 1 1 1 0 0 0
...	...

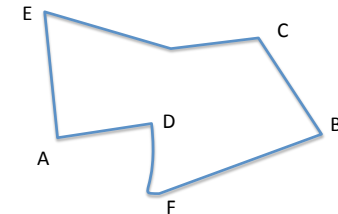
Chromosome 1	1 5 3 6 0 1 2 7 3 0 8
Chromosome 2	9 2 4 1 8 3 2 6 2 1 0
...	...



Representation / chromosome encoding

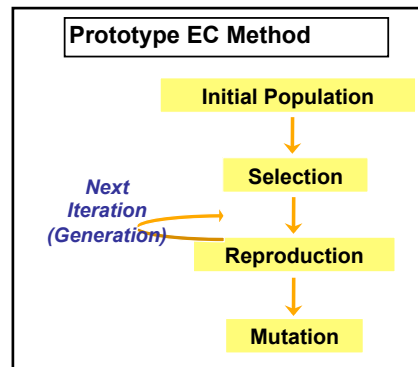
- Domain dependent
- Essential ingredient

Exercise: Propose a representation for circuits in the travelling salesman problem



Functioning of the Genetic Algorithm

- Basic cycle



Initialization of the population

- N individuals generally randomly generated
- N is domain-dependent
 - Often in [~ 50 - ~ 1000]

Fitness function

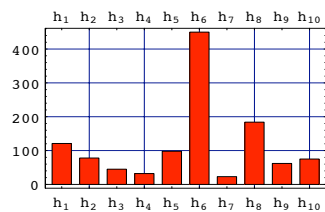
- Evaluates the quality of the solution
 - E.g. *z-value* in function optimization
 - *Length of the circuit* in the travelling salesman problem
 - *Time before falling down* in the inverse pole
- Beware of its cost
 - Keep values in memory

Selection

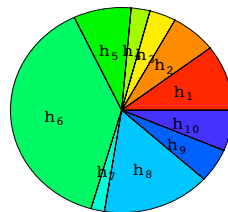
- Idea: **keeping the most promising parents for generating the next population**
 - Keep best individuals according to the fitness function
 - While preserving genetic diversity
- Several selection methods
 - Elitist
 - Roulette-wheel
 - Tournament
 - Rank
 - ...

Selection: Roulette-wheel

- The **probability** of selecting an individual is **proportional to its fitness**



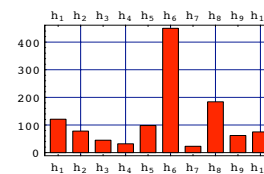
Fitness



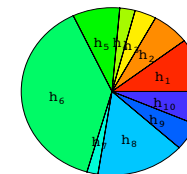
Probability of selection

Selection: rank

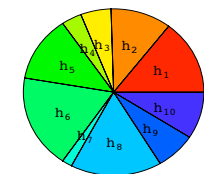
- The **probability** of selecting an individual is **proportional to its rank**



Fitness



Probability of selection according to **fitness**



Probability of selection according to **rank**

Selective pressure is softened

Selection: Tournament

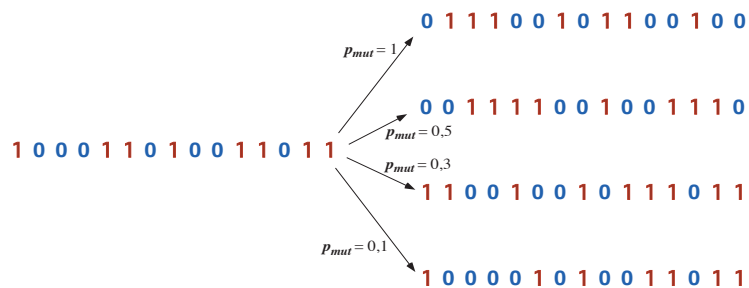
- **Selection by fitness or rank** implies the evaluation of the fitness of all individuals
- **Selection by tournament** avoids this
 - If n individuals must be selected (within a population of size N)
 - Organize n tournaments, each between $m < N$ randomly chosen individuals (m controls the selective pressure)
 - Select the best individual / or select the best and second best / or ...

- Generally less evaluations
- Less sensitivity to errors of the fitness function

Genetic operators

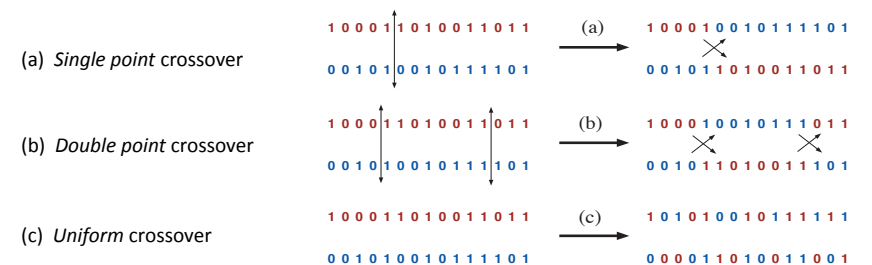
- **Operators on chromosomes** used to renew the population and explore the space of solutions
- **Large variety**
 - Holland suggested to mimic natural evolution with
 - **Mutation**
 - **Crossover**
- Often difficult to assess the merits of the proposed operators

Genetic operators: mutation



Mutation operator with various degrees of probability

Genetic operators: crossover



Genetic operators: role

- Assure trade-off between
 - **Exploitation**
 - Preserve best individuals and explore nearby locations
 - **Mutation** is exploitation oriented
 - Small steps but brings new alleles
 - **Exploration**
 - Search unexplored regions for possible good candidates
 - **Crossover** is exploration oriented
 - Large steps but does not bring new alleles

16/12/10

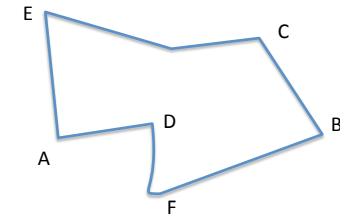
Evolutionary Computation - © A. Cornuéjols

21

Genetic operators

Exercise:

- Propose a **representation** for circuits in the travelling salesman problem
- And the corresponding **mutation** and **crossover operators**



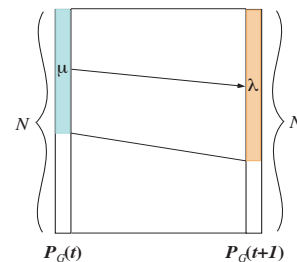
16/12/10

Evolutionary Computation - © A. Cornuéjols

22

Replacement of the population

- Selection of μ parents
 - By fitness / rank / tournament / ...
- Generation of λ children
 - Mutation / crossover / copy
 - And selection of the best
- Completion to N
 - Elimination of the worst individuals and copy of others



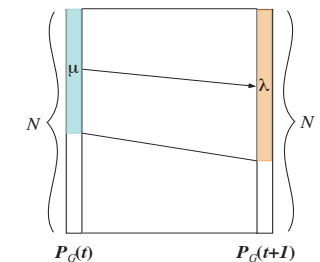
16/12/10

Evolutionary Computation - © A. Cornuéjols

23

Replacement of the population: 3 strategies

1. Completely replace the previous population (called **(μ, λ) replacement**)
 - Risk: losing the good individuals of previous population
2. Draw the N new individuals from the selected μ parents and λ children (called **$(\mu + \lambda)$ replacement**)
3. **Steady state**
 - Select a sub-population and make replacement for this sub-population only (possibility of parallel and asynchronous process)



16/12/10

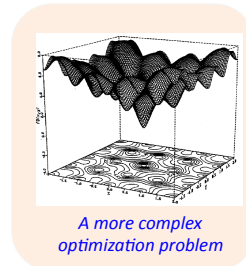
Evolutionary Computation - © A. Cornuéjols

24

Simple GA: example

(from Goldberg, 1989)

- Problem: finding Argmax of x^2 over $\{0, \dots, 31\}$
- **GA approach**
 - **Representation:** binary code (e.g. 0 1 1 0 1 \leftrightarrow 13)
 - Population size = 4
 - **Operators**
 - Single-point crossover
 - Mutation
 - Roulette wheel **selection** according to fitness
 - **Random initialization** of the population



16/12/10

Evolutionary Computation - © A. Cornuéjols

25

Simple GA: example

(from Goldberg, 1989)

- **Selection**

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

16/12/10

Evolutionary Computation - © A. Cornuéjols

26

Simple GA: example

(from Goldberg, 1989)

- **Crossover**

String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 1	4	0 1 1 0 0	12	144
2	1 1 0 0 0	4	1 1 0 0 1	25	625
2	1 1 0 0 0	2	1 1 0 1 1	27	729
4	1 0 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

16/12/10

Evolutionary Computation - © A. Cornuéjols

27

Simple GA: example

(from Goldberg, 1989)

- **Mutation**

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

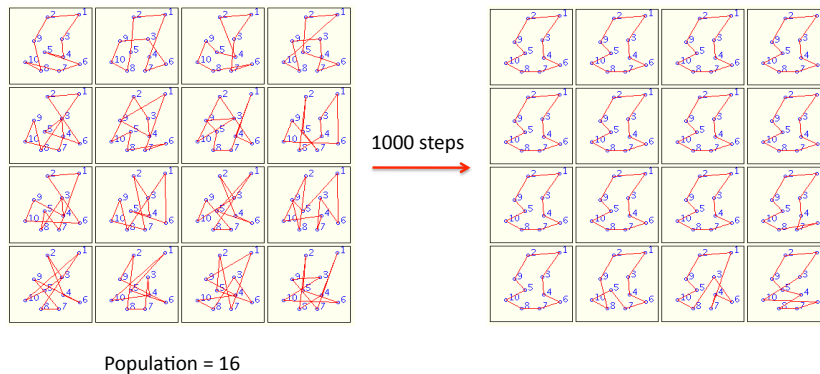
16/12/10

Evolutionary Computation - © A. Cornuéjols

28

Illustration: Traveling Salesman Problem

<http://www.obitko.com/tutorials/genetic-algorithms/tsp-example.php>



16/12/10

Evolutionary Computation - © A. Cornuéjols

29

Why GAs work: the schema hypothesis

- Analysis of the dynamical behavior [Holland, 1975]
- *“GAs work by discovering, emphasizing and recombining good ‘building blocks’ of solutions in a highly parallel fashion”*
 - Melanie Mitchell, paraphrasing John Holland –
- **Schema** = building block
- Schemas are propagated or destroyed according to the laws of probability

16/12/10

Evolutionary Computation - © A. Cornuéjols

31

Lessons

- **General purpose** optimization method
 - Try when no canonical method work
 - (e.g. discontinuous domain, complex fitness)
 - No guarantee on the goodness of the solution
- **Often slow**
 - Need to set various parameters
 - **Encoding** of the solutions
 - **Population size**
 - **Reproduction operators**
 - **Fitness definition** (avoid costly one if possible)

16/12/10

Evolutionary Computation - © A. Cornuéjols

30

Why GAs work: the schema hypothesis

- GAs evaluate and operate explicitly on solutions
- But GAs implicitly evaluate and operate on schemas
 - Schemas may be destroyed or weakened by crossover and mutation
 - New schemas may be spliced together from existing schemas
- Good schemas are associated with solutions of high fitness
- Fitter offspring are probabilistically more likely to be chosen to reproduce (thereby propagating good schemas)

16/12/10

Evolutionary Computation - © A. Cornuéjols

32

Why GAs work: the schema hypothesis

- Schemas
 - E.g. : $11***01*1*$ implicitly represents 2^5 solutions
- Length : Distance between the two most distant bits ($\neq*$)
 - E.g. length $d(11***01*1*) = 8$
 - length $d(*1***0****) = 4$
- Order: Number of bits ($\neq*$)
 - E.g. order $o(11***01*1*) = 5$
 - order $o(*1***0****) = 2$

16/12/10

Evolutionary Computation - © A. Cornuéjols

33

Why GAs work: the schema hypothesis

- Let H be a schema with at least one instance present in the population at time t
- Let $m(H, t)$ be the number of instances of H at time t
- Let x be an instance of H and $f(x)$ be its fitness
- The expected number of offspring of x is $f(x)/f(pop)$ (by fitness proportionate selection)

Goal: compute $E\{m(H, t+1)\}$ (the expected number of instances of schema H at time $t+1$)

16/12/10

Evolutionary Computation - © A. Cornuéjols

34

Why GAs work: the schema hypothesis

- Schemas of **high length** are less likely to survive **crossover**
- Schemas of **high order** are less likely to survive **mutation**

$$E[m(s, t + 1)] = \frac{\phi(H, t)}{\bar{\phi}(t)} m(H, t) \left(1 - P_{crossover} \frac{d(H)}{L - 1}\right) (1 - P_{mut})^{o(H)}$$

This theorem has limits in its explicative power

But it introduces the great idea of implicit exploration of the search space

16/12/10

Evolutionary Computation - © A. Cornuéjols

35

Conclusions

- A very general purpose approach
 - Optimization
 - Problem-solving
 - Co-evolution
 - Machine Learning
- Still incompletely understood

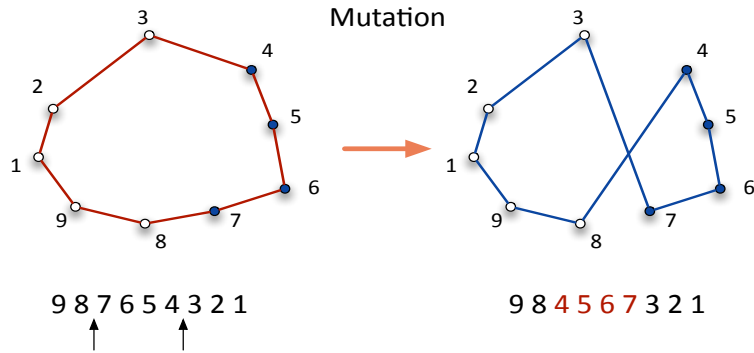
16/12/10

Evolutionary Computation - © A. Cornuéjols

36

The Traveling Salesman Problem

- A solution: the “2-opt mutation”



References

- Eiben, A. and Smith, J. *Introduction to Evolutionary Computing*. Springer, 2007.
- Goldberg, David. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
- Holland, John. *Adaptation in Nature and Artificial Systems*. University of Michigan Press, 1975.
- Koza, John. *Genetic Programming: On the programming of computers by means of natural selection*. MIT Press, 1992.
- Mitchell, Melanie. *An Introduction to Genetic Algorithms*, MIT Press, 1997.
- **Conferences**
 - GECCO (The Genetic and Evolutionary Computation Conference)