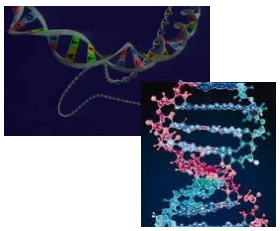
§ 4.3 Evolutionary Computation

- Evolutionary computation
- Genetic algorithm theory
- Genetic encoding



Evolution

- Evolution is the change in the inherited traits of a population from one generation to the next

- Natural selection leading to better and better species

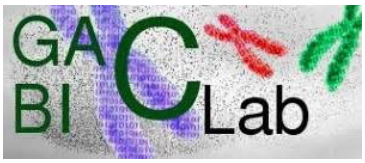
Evolution – Fundamental Laws

- Survival of the fittest.
- Change in species is due to change in genes over reproduction or/and due to mutation

An example showing the concept of survival of the fittest and reproduction over generations

1. Evolutionary Computation

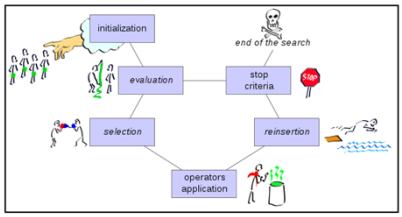
- A subfield of artificial intelligence which mimics biology
- Used in optimization of black box problems
- Parallel processing
- Biologically inspired algorithms



What is Evolutionary Computation

EC refers to computer-based problem solving systems that use computational models of evolutionary process.

- EAs fall into the category of "*generate and test*" algorithms
- They are stochastic, population-based algorithms
  - no proven upper bound for run time
  - optimal solution not guaranteed
  - no guarantee for any solution quality



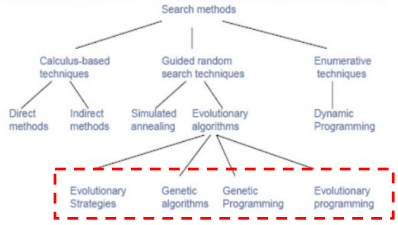
General schema of an Evolutionary Algorithm (EA)

Types of EC

Types of Evolutionary Computation

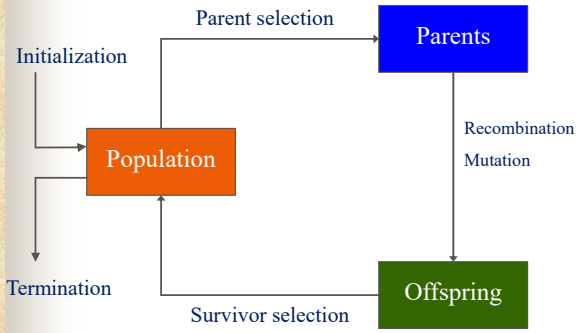
- Evolutionary programing
- Genetic algorithm
- Evolutionary strategy
- Genetic programing

- Genetic Algorithm
- Swarm Intelligence



General frame

Evolutionary Computation General Frame



Genetic algorithm

1. Genetic algorithm

What is a Genetic Algorithm (GA)

-*Genetic algorithms* are defined as global optimization procedures that use an analogy of genetic evolution of biological organisms

-Basically genetic algorithms tend to find the best solution to a problem by following an evolutionary process

What is gene

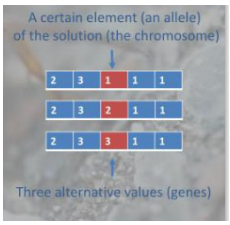
Genetic algorithm--what is gene ?

Biology:

A certain DNA sequence at a certain position of the chromosome

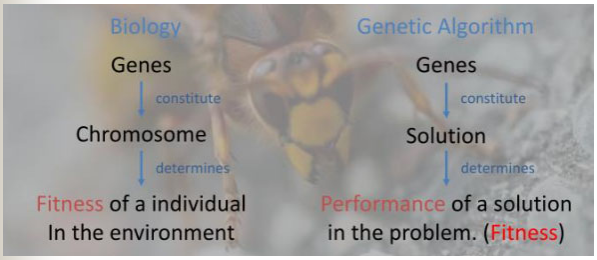
Genetic algorithm:

A certain value of a certain element of the solution



What is gene

Genetic algorithm--what is gene ?



What is gene

Genetic algorithm — what is gene ?

In GA, *genes* (values of elements of the solution) determine the *fitness* (performance) of a solution.

To solve a problem

=

To find the *combination of genes*

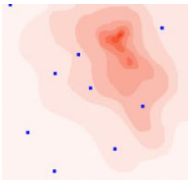
that provides *the best* fitness (performance)

Initiation

Genetic algorithm--Initiation

-To conduct evolution, we need a *set of solutions* (A *population*)

-Initially, the *population* is generated *randomly*. This the first generation.

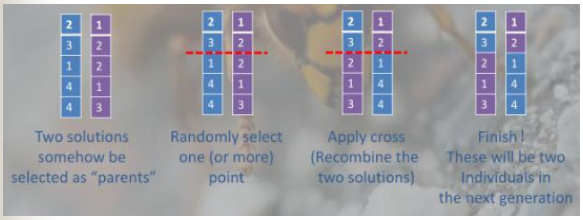


A two-dimension search space dotted by randomly generated solutions (each solution consists of two elements, x and y)

Crossover

Genetic algorithm—Reproduction: *Crossover*

Crossover is how we *create new individuals* from the existing ones.



Crossover

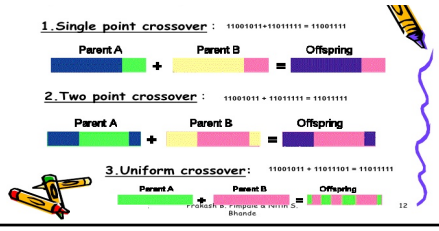
Genetic algorithm—Reproduction: *Crossover*

Generating offspring from two selected parents

Single point crossover

Two point crossover (Multi point crossover)

Uniform crossover

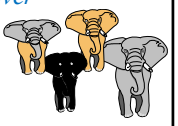


Crossover

Genetic algorithm—Reproduction: *Crossover*

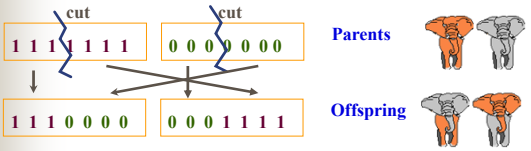
Crossover for discrete representation

Whole population:



Each chromosome is cut into *n* pieces randomly

E.g. *Single point crossover* random position is 3



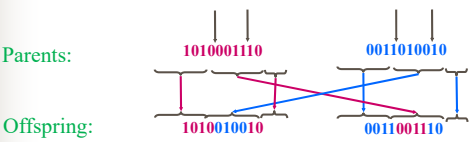
Crossover

Genetic algorithm—Reproduction: *Crossover*

Example for *n*=2, *Two point* crossover

Randomly two positions in the chromosomes are chosen

Randomly chosen positions



Uniform crossover

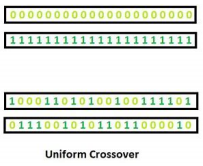
Genetic algorithm—Reproduction: *Uniform crossover*

- A random mask is generated
- The mask determines which bits are copied from one parent and which from the other parent
- Bit density in mask determines how much material is taken from the other parent

Mask: 0110011000 (Randomly generated)

Parents: 1010001110 0011010010

Offspring: 0011001010 1010010110



Uniform crossover

Genetic algorithm—Reproduction: *Uniform crossover*

Problems with crossover

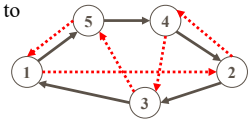
Depending on coding, simple crossovers can have high chance to produce *illegal offspring*

Example

TSP

Route A: 1 5 4 2 3

Route B: 1 2 4 3 5



1 5 4 2 3

1 2 4 3 5

Doesn't work!!!

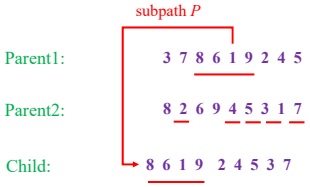
Uniform crossover

Uniform crossover can often be modified to avoid this problem

- E.g. in TSP with simple path coding:

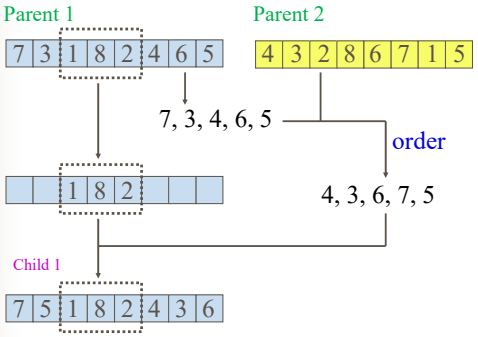
  -Select a random subpath *P* from parent 1

  -Turn the subpath *P* into a complete tour by visiting the cities not in *P* in the order they appear in parent 2



Another way

Genetic algorithm—Reproduction: *Another crossover*

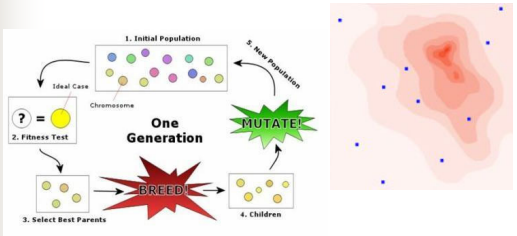


Selection

Genetic algorithm—Reproduction: *Selection*

-Individuals with *higher fitness* have a *higher probability* to be chosen as parents of the *crossover* operation.

-Survival of the fittest



Selection

Genetic algorithm—Reproduction: *Selection*

What's the effect ?

Genes associated with *high fitness* are *more likely* to be passed to the new generation.

After some generations, the *average fitness* of the population gets *improved*.

Selection strategy

Genetic algorithm—Reproduction: *Selection strategy*

Most important and computationally expensive step of a GA

- Ensure better individuals have a better chance
- Drive the population forward
- Be careful
  - to give less good individuals at least some chance of being parents - they may include some useful genetic material
- Typical ways
  - Roulette wheel selection
  - Tournament selection

Roulette wheel

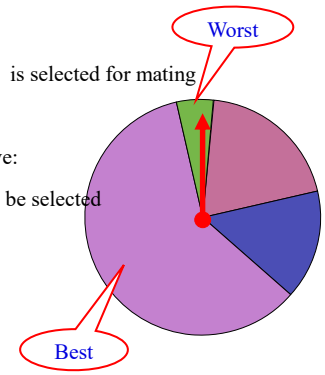
Genetic algorithm—Reproduction: *Selection strategy*

Roulette wheel selection

-Expected number of times f_i is selected for mating is: $f_i / \sum f_j$

-Better (fitter) individuals have:

more space and chances to be selected



Roulette wheel

Genetic algorithm—Reproduction: *Selection strategy*

Roulette wheel selection

- Sum the fitness of all chromosomes, call it T
- Generate a random number N between 0 and T
- Return chromosome whose fitness added to the running total is equal to or larger than N
- Chance to be selected is exactly proportional to fitness

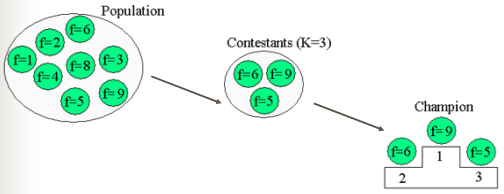
Chromosome :	**1**	**2**	**3**	**4**	**5**	**6**
Fitness:	**8**	**2**	**17**	**7**	**4**	**11**
Running total:	**8**	**10**	**27**	**34**	**38**	**49**
N (1 ≤ N ≤ 49):			**23**			
Selected:			**3**			

Tournament

Genetic algorithm—Reproduction: *Selection strategy*

Tournament selection

Select ***k*** random individuals, take the best ***k*** is called the size of the tournament



Selection

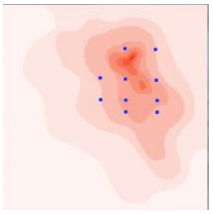
Genetic algorithm—Reproduction: *Selection*

In a graphic view: (use our two-dimension example).

The population gathers around the *optimal solution*.

It's like that the population is climbing the hill

Problem solved ?



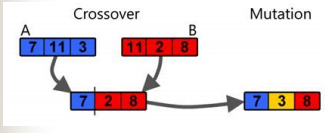
Selection

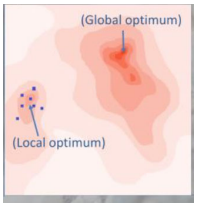
Genetic algorithm—*Selection*

Problem: What if we have multiple hills in the searching space?

The individuals may climb onto a hill that is not the highest.

Thus, they may gather around a *local optimum*.





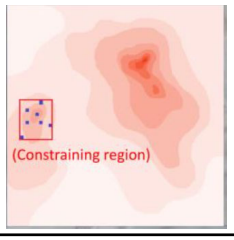
Selection

Genetic algorithm—*Selection*

According to the *crossover* operation, genes in the new generation only come from the previous generation.

Thus, once the solutions gather around a *local optimum*, they will be *constrained* in its vicinity!

They won't find the *global optimum*.



Mutation

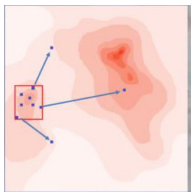
Genetic algorithm—*Mutation*

Mutation: Make random changes to some genes in each generation.

New genes are created!

Solutions can *jump out* of the region.

After some generations, they may probably gather around the *global optimum*



Mutation

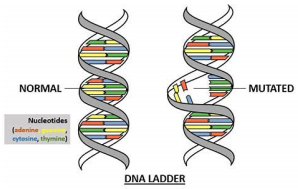
Genetic algorithm—*Mutation*

Mutation for Discrete Representation

before 1 1 1 1 1 1 1

after 1 1 1 0 1 1 1

mutated gene



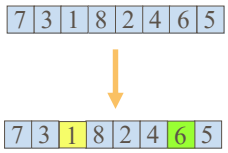
Mutation usually happens with probability p_m for each gene

Mutation

Genetic algorithm—*Mutation*

Mutation for order based representation (*Swap*)

-Randomly select two different genes and swap them



Scenario

Genetic algorithm—*Scenario*

Step 1: Initiation (Randomly generate the first generation)

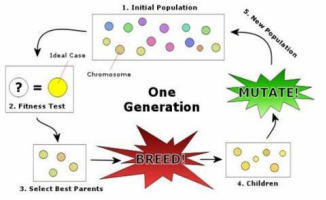
Step 2: Mutation

Step 3: Fitness evaluation

Step 4: Reproduction

Selection

Crossover



Step 5: Go back to Step 2, repeat this loop until a sufficiently good solution is found.

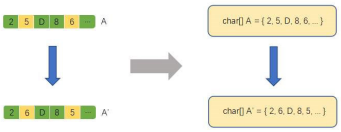
Encoding

3. Genetic algorithm--*Encoding*

- All kind of *alphabets* can be used for a chromosome (numbers, characters), generally a binary alphabet is used

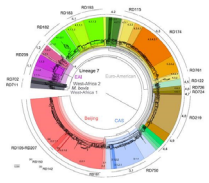
Coding

- Chromosomes could be:
  - Bit strings (0101 ... 1100)
  - Real numbers (43.2 -33.1 ... 0.0 89.2)
  - Permutations of element (E_{11} E_3 E_7 ... E_1 E_{15})
  - Lists of rules (R_1 R_2 R_3 ... R_{22} R_{23})
  - Program elements (genetic programming)
  - ... any data structure ...



Coding

- Order of genes on chromosome can be important
- Generally many different coding for the parameters of a solution are possible
- Good coding is probably the most important factor for the performance of a GA

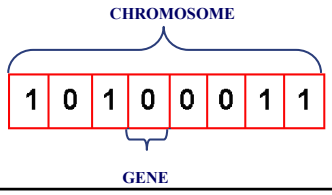


Example

Example: Discrete representation (Binary alphabet)

-Representation can be using discrete values , such as

- Binary
- Integer
- Any other system with a discrete set of values.



Example

Example: Discrete representation (Binary alphabet)

8 bits Genotype

1 0 1 0 0 0 1 1

Phenotype:

- Integer
- Real Number
- Schedule
- ...
- Anything?

Example

- Example: Discrete Representation (Binary alphabet)
  - Phenotype could be integer numbers

Genotype: 1 0 1 0 0 0 1 1 Phenotype: = 163

$1*2^7 + 0*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 =$

$128 + 32 + 2 + 1 = 163$

Example

Example: Discrete representation (Binary alphabet)

-Phenotype could be real numbers

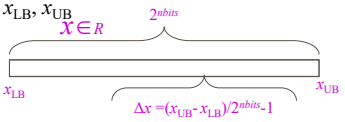
-E.g. a number between 2.5 and 20.5 using 8 binary digits

Genotype: 1 0 1 0 0 0 1 1 Phenotype: = 13.9609

$$x = 2.5 + \frac{163}{256}(20.5 - 2.5) = 13.9609$$

Coding

- Binary Encoding Issues
  - Resolution depends on:
    - upper and lower bounds x_{LB}, x_{UB}
    - number of bits



Encoding: Example

x is a real number within [5,10], and the precision is 10^{-5}, so the closed interval must be divided into 50000 parts at least, since $262144=2^{18}<500000<2^{19}= 524288$, so

nbits=19, $\Delta x=5/2^{19}$

0000000 … 0000000=0 →5

0000000 … 0000001=1 →5 + Δx

…

1111111 … 1111111=2^{19}-1 →10

Decoding

Genetic algorithm—*Decoding*

For X: $x_l x_{l-1} \ldots x_2 x_1$

$$x = x_{LB} + \Delta x \times \sum x_i 2^{i-1}$$

Example

[G]=encode(137.56, 50, 150, 8)

G = 1 1 0 1 1 1 1 1

[X]=decode(G, 50, 150, 8);

X = 137.4510

So, $\Delta x = (150-50)/2^8 = 0.39$

Loss in precision !!!?

Example: Discrete Representation (Binary alphabet)

- Phenotype could be a Schedule
- E.g. 8 jobs, 2 time steps

Genotype:

1	0	1	0	0	0	1	1

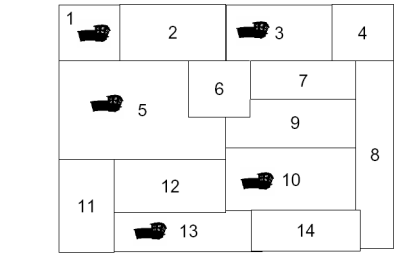
=

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

Phenotype

Example: Discrete Representation (Binary alphabet)

- A representation for the fire station location problem



1 0 1 0 1 0 0 0 0 1 0 0 1 0

"1" represents a fire station

Other coding

Genetic algorithm—*Other coding*

- Example: Real number representation
- Real number table

A very natural encoding if the solution we are looking for is a list of real-valued numbers

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

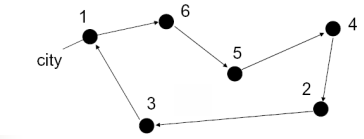
The fitness function maps tuples of real numbers to a single real number

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

Example

Example:

TSP task, two coding schemes



Arc

1 6 5 4 2 3

6 5 4 2 3 1

Orders

1 6 5 4 2 3

Same problem, but two different chromosome representations

Limitation

Genetic algorithm—*Limitations*

- Fitness value is calculated repeatedly which might be computationally expensive for some problems.
- Being stochastic, there are no guarantees on the optimality or the quality of the solution.
- If not implemented properly, the GA may not converge to the optimal solution.