5– Search Methods
Genetic Algorithms

Carlos Ramos
csr@isep.ipp.pt

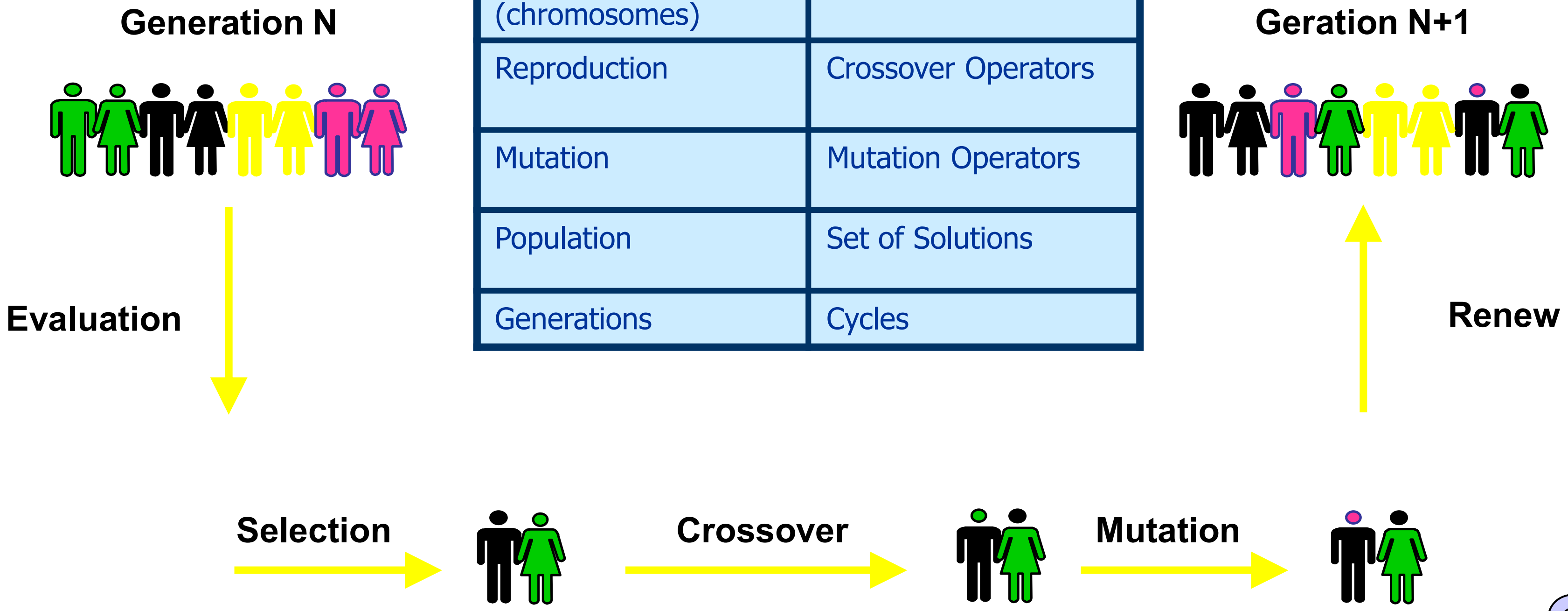
Curricular Unit: Advanced Algorithmics
Programme: BSc on Informatics Engineering (ISEP)

Genetic Algorithms

- In the early 1970's John Holland developed research to see if aspects of natural evolution could be incorporated into algorithms that allow automatic problem solving.
- According to John Holland, each solution to a problem could be understood as an individual in a population, so all individuals in the population would be equivalent to a set of possible solutions to a problem, each individual would be represented by their genotype, that is, individuals would be represented by chromosomes
- Such populations would generally evolve based on the reproduction process, assuming that occasional mutations could occur.
- In this way the population was renewing itself through generations, which would correspond cycles of the algorithm
- The process was repeated until we reached an individual (solution) that presented the minimum desired characteristics or until a time limit imposed the end of the algorithm

Genetic Algorithms

Aspects of natural evolution can be incorporated into algorithms that allow automatic problem solving



Natural Evolution	*Genetic Algorithms*
Individual	Solution
Genotype (chromosomes)	Representation
Reproduction	Crossover Operators
Mutation	Mutation Operators
Population	Set of Solutions
Generations	Cycles

Genetic Algorithms

- The crossing operator combines the characteristics of two individuals (progenitors) in order to generate two new individuals (descendants), obtained by composing the genetic material of their parents.
- The mutation operator consists of the arbitrary alteration of 1 or more genes of a chromosome with an arbitrary probability. This operator ensures that there is some genetic diversity in the population and aims to prevent premature convergence to optimal locations by randomly sampling new points in the search space.
- By cyclically crossing and mutating and selecting the "individuals" who go to the next generation, we implement the Genetic Algorithms

Crossover Operator

- The main operator of Genetic Algorithms is the crossover operator
- Two parent individuals are chosen from among the population by a well-defined selection method
- When chromosomes are represented only by sequences (eg, bits, numbers or characters), the crossover operator generally produces the two offspring by choosing one or more cutoff points on the parent chromosomes and then mixing the resulting parts to generate each of the descendants' chromosomes
- Alternatively, offspring can be generated by masks that indicate whether the gene in each position should be taken from one parent or the other.
- There are therefore 3 basic crossover alternatives:

  crossing at a cutoff

  two-point crossover

  uniform crossover (with mask)

Crossover Operator

- Crossover operator randomly selects a cut position
- The descendant chromosome D1 is obtained by the part of the chromosome that is before the cutoff point in parent P1 and by the part of the chromosome of progenitor P2 that is after the cutoff point.
- The descendant chromosome D2 is obtained by the part of the chromosome that is before the cutoff point in parent P2 and by the part of the chromosome of progenitor P1 that is after the cutoff point

Crossover Operator

Let’s see na example considering the binary codification for the chromossomes of parents P1 e P2:

P1 = 10110010

P2 = 11010110

If we consider the cutting point between the 5th and 6th genes (or bits, from left to right) then the parents’ chromossomes are in the following way:

P1 = 10110-010

P2 = 11010-110

Thus descendants are as follows:

D1= 10110-110 = 10110110

D2= 11010-010 = 11010010.

Crossover Operator

- The crossover operator at two cutoff points randomly selects two cutoff points by dividing each parent's chromosomes into 3 parts
- The first descendant chromosome is obtained by the extreme parts of the first parent chromosome and the central part of the second parent chromosome
- The second descendant chromosome is obtained by the extreme parts of the second parent chromosome and the central part of the first parent chromosome

Crossover Operator

Let’s consider P1 and P2 chromossomes again:

P1 = 10110010

P2 = 11010110

Let’s suppose that the cutting points are between 2nd e 3rd genes and between 5th and 6th genes, thus we have:

P1 = 10-110-010

P2 = 11-010-110

Now the descendants are obtained as follows:

D1 = 10-010-010 = 10010010

D2 = 11-110-110 = 11110110.

Crossover Operator

- We could extend the previous concept to more cut points (3, 4, etc.)
- This is possible with the uniform crossover operator that will randomly generate the positions of the genes to be exchanged, which corresponds to generate a binary mask of the same size as the chromosomes, meaning 1 that the gene should be exchanged at the respective position and 0 that the gene should not be exchanged

Crossover Operator

Let’s take P1 and P2 again:

P1 = 10110010

P2 = 11010110

Assume that the randomly-generated mask is:

M = 01110110

Thus the descendants are:

D1 = 1**1010**0**11**0

D2 = 1**0110**1**01**0

Crossover Operator

- Another crossing operator is the Order Crossover

- Let's consider the following problem: We have a set of 9 tasks that can be performed in any order on a machine. The evaluation function could have to do with processing time optimization issues considering tool change lead times, but this is just a minor detail for the present case

- Chromosome encoding could be done by 9-digit sequences of the following alphabet: {1,2,3,4,5,6,7,8,9}

Crossover Operator

Two potential parents could be:

Pa = 247139865

Pb = 753218649

If, for instance, we adopt the 2-point crossover(between the 2nd and 3rd genes and between the 6th and 7th genes) we will have the following:

Pa = 24-7139-865

Pb = 75-3218-649

If we made the crossover with 2 cutting points we will have the following descendants:

Da = 24-3218-865 = 243218865

Db = 75-7139-649 = 757139649

Notice we have now repeated values and values missing, corresponding to non-feasable solutions

Crossover Operator

- For this reason, crossover operators appear as the Order Crossover operator where the genes to be replaced from one parent will appear in the order in which they are in the other parent. This exchange may take place from the first or second cutoff point.

- Considering L as the location of a free gene and assuming that the central part remains, we have for the example cited the following sequences:
  - Sequence obtained from Pa = LL-7139-LLL
  - Sequence obtained from Pb = LL-3218-LLL

Crossover Operator

Sequence from 2nd cutoff point Pb = 649753218

Sequence from the second cutoff point of Pa = 865247139

We will now eliminate the known elements in the central part of the chromosome of Pa (7139) in sequence from the 2nd cutoff point of Pb and the known elements in the central part of the chromosome of Pb (3218) in sequence from the 2nd cutoff point. Pa, giving rise to the following sequences

Sequence from 2nd Pb cutoff eliminating 7139 = 649753218 = 64528

Sequence from 2nd Pa cutoff eliminating 3218 = 865247139 = 65479

Such sequences will fill in the specified order the previously mentioned free positions (L) from the second cutoff point, giving rise to the following descendants Da and Db:

Da = 28-7139-645 = 287139645

Db = 79-3218-654 = 793218654

Mutation Operation

- The mutation operator used in Genetic Algorithms aims to bring back to the population the genes lost during the selection process so that they can be tested in a new context. It also serves to provide new genes that were not originally in the initial population.
- Generally, the mutation is performed on only 1 gene, although it can be done on more genes. At the randomly chosen position the chromosome gene is replaced by another viable gene for that position, derived from the population's genetic heritage (set of all genes that may appear in the population).

Mutation Operation

Consider, for example, descendant D:

D = 10110010

If the chromossome of this descendant has a mutation in the 5th gene, having a 0, the gene is substituted by 1. So the new version of the descendant is:

D′ = 1011**1**010

Mutation Operation

Consider now the descendant Da:

Da = 287139645

If we applied the mutation to a given gene (for example, the 2nd gene that has the value 8) and originated the set of genes that could appear in that position we would get the following set: {1,2,3,4,5,6, 7.9}.

Note that mutation of the 2nd position gene would always result in a non-viable solution, where 8 would not appear and a task would be repeated.

For this reason the mutation here should be performed on 2 genes, exchanging their contents. For example, if we change the 2nd and 5th Da gene we will get the following mutation:

Da '= 2**3**71**8**9645

Genetic Algorithm

A genetic algorithm for a given problem must contain the following components:

- a genetic representation for potential problem solutions
- a way to generate an initial population of potential solutions
- an evaluation function that plays the role of the environment, classifying solutions according to their suitability
- genetic operators that alter the composition of offspring
- values for the various parameters that the genetic algorithm uses (population size, probabilities of application of genetic operators, etc)

Genetic Algorithm

The following algorithm summarizes the operation of a typical genetic algorithm:

```
Generate initial population of individuals
While the stopping criterion is not met
    Assess the fitness of all individuals
    Select the most suitable individuals for reproduction
    Generate new individuals through crossover and assess their
      fitness
    Apply mutation to some individuals
    Generate new population by inserting some good individuals
      and eliminating bad individuals
End
```

Example – Maximum of a function

- This first example is simple, but will allow us to see how a genetic algorithm can evolve
- Let's consider the function f (x) = x^2, with x integer and belonging to the range [0,63]. The aim will be to maximize the value of the respective function
- Encoding solutions by a string of 6 binary digits is appropriate
- Let's consider the following table that gives us the population elements and the value of the evaluation function (equal to f (x))

Example – Maximum of a function

Element n.	Generation 1	Solution x	f(x)
1	100101	37	1369
2	001001	9	81
3	010011	19	361
4	001100	12	144

Example – Maximum of a function

After the first crossover:

Element n.	Population after crossover	Solution x	f(x)
1	100101	37	1369
2	001001	9	81
3	010011	19	361
4	001100	12	144
5 - cruz.1,2	100001	33	1089
6 - cruz.1,2	001101	13	169
7 - cruz.3,4	000110	6	36
8 - cruz.3,4	011001	25	625

Let’s use now the uniform crossover operator with the mask M = 010101

Example – Maximum of a function

So, elements 1, 3, 5 and 8 are selected and renumbered, creating Generation 2 as follows:

Element n.	Generation 2	Solution x	f(x)
1	100101	37	1369
2	010011	19	361
3	100001	33	1089
4	011001	25	625

Example – Maximum of a function

We will repeat the process one more time:

Element nº	Population after crossover	Solution x	f(x)
1	100101	37	1369
2	010011	19	361
3	100001	33	1089
4	011001	25	625
5 - cruz.1,2	110001	49	2401
6 - cruz.1,2	000111	7	49
7 - cruz.3,4	110001	49	2401
8 - cruz.3,4	001001	9	81

Example – Maximum of a function

Now elements 1, 3, 4 and 5 are selected giving origin to Generation n. 3:

Element n.	Generation 3	Solution x	f(x)
1	100101	37	1369
2	100001	33	1089
3	011001	25	625
4	110001	49	2401

Example – Maximum of a function

We will not continue to exemplify the following generations. Let's just look at a few things:

- Firstly, after 2 cycles of crossing it was found that the population has higher rating factors than the original population and that the best element of the population has a higher rating factor
- Does that mean that we will quickly converge on the optimal solution?
  - Unfortunately, this is not the case. If we continue to apply only crossings from generation 3 onwards, we can even guarantee that the best solution will never be found. The best solution is 111111 and note that in the 5th bit, from left to right, all population elements have the value 0

Example of task sequencing in a machine

- Consider sequencing 9 tasks {1,2,3,4,5,6,7,8,9} on a machine, where odd tasks are performed by a fi tool and even tasks are performed with an fp tool . The goal is to minimize the number of tool changes
- The coding will be done by a sequence of 9 digits and the evaluation function will try to minimize the number of tool changes, that is, it will count the number of even to odd and odd to even passes in the sequence
- It is advisable to use an order-keeping crossover operator, as the other 3 crossover operators would result in non-viable solutions (with repeated tasks and missing tasks).

Example of task sequencing in a machine

Element n.	Generation 1	f(x)
1	394165287	6
2	821397465	3
3	136974285	4
4	754381296	7

Let's apply the crossover operator, the order-keeping crossover operator and 2 cutoff points (between the 2nd and 3rd genes and between the 6th and 7th genes). Again we will cross element 1 with 2 and element 3 with 4

Example of task sequencing in a machine

After crossover we have:

Element n.	Population after Crossover	f(x)
1	39-4165-287	6
2	82-1397-465	3
3	13-6974-285	4
4	75-4381-296	7
5 - cruz.1,2	97-4165-823	6
6 - cruz.1,2	65-1397-284	2
7 - cruz.3,4	81-6974-253	5
8 - cruz.3,4	97-4381-256	7

Example of task sequencing in a machine

By elitism we obtain:

Element n.	Generation 2	f(x)
1	821397465	3
2	136974285	4
3	651397284	2
4	816974253	5

Parametrization of a Genetic Algorithm

The main parameters of a Genetic Algorithm are:

- population size
- the crossing rate
- the mutation rate
- the replacement rate
- the stopping criteria

Parametrization of a Genetic Algorithm

Population size determines the quality of the solution obtained and the processing time

- If the population is too small, we are likely to have a poor diversity of solutions that can be obtained in the search space, and often a situation is quickly reached where the population is not improved, unless mutations casually give rise to more interesting individuals
- If the population is very large, we have better coverage of the search space, with less probability of the population stabilizing a local high. However, this is achieved with computational costs
- Genetic algorithms generally assume that the population remains constant in terms of number of individuals, although nothing prevents their dynamic variation, although in practical terms care should be taken to avoid abrupt population decline or growth

Parametrization of a Genetic Algorithm

Crossing Rate will define the probability that 1 individual will cross to generate new offspring

- A low Crossover Rate causes generations of the population to be poorly renewed, making evolution very slow
- A high crossover rate leads to a major renewal from generation to generation, which, on the one hand, may be positive to avoid the problem described above, may also lead to the need for replacement of individuals with high fitness
- A typical crossover rate is between 60% and 80%

Parametrization of a Genetic Algorithm

Mutation is an important operation since it can generate better able individuals, thereby evading local maxima of the problem. With mutation we can introduce an additional factor of diversity into the population

- A very high mutation rate leads to a random search that may not lead to convergence, as a mutation is most likely even generating poorer individuals.
- Typical mutation rate values are quite low (eg about 1%)

Parametrization of a Genetic Algorithm

The Replacement Rate defines the percentage of population that will be replaced at each generation transition.

- Too high a value may lead to excessive population renewal while too low a population stagnation

Parametrization of a Genetic Algorithm

Generally one of the following stopping criteria is adopted:

- when a solution with a certain suitability value is reached (for the type of problem where a solution with a cost less than a given value, but no need to find the best solution is sought)
- when a certain time limit is reached (for problems that require timely or real-time solutions)
- when it reaches a certain number of generations

Genetic Algorithms Evaluation and Fitness

- The notions of evaluation and fitness are sometimes interchangeable. However, one should distinguish the notion of evaluation function and fitness function used in Genetic Algorithms
- Evaluation provides a measure of performance against a particular set of parameters, and the evaluation of a given individual is independent of other individuals
- Fitness transforms this measure of performance into the attribution of survival and reproduction opportunities, and is therefore defined in relation to other individuals in the population
- Calculating evaluation and fitness functions should not be too complex and time consuming

Selection

- Choosing the most adapted individuals from the population through a deterministic selection operator may seem logical
- Such choice, said elitist, can, however, lead to the reduction of population diversity, making a good solution obtained in the beginning become dominant and avoiding the proper exploration of the search space.
- It is important that the least adapted individuals are likely, even if small, to be selected

Selection

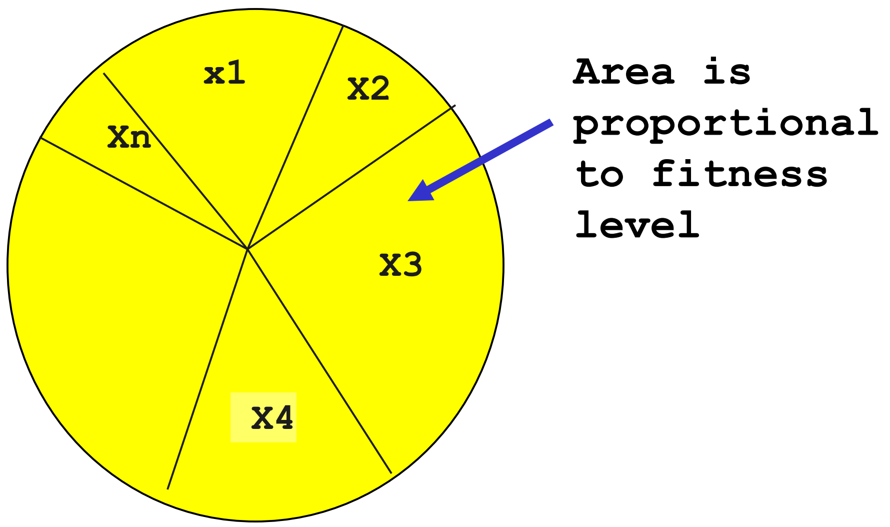
One of the most common proportional selection methods is the Roulette Wheel Parent Selection method

- whereby each individual i in the population is given a probability of selection Ps (i), calculated by the quotient between his evaluation and the sum of the evaluations of all individuals in the population, or by a similar formula (sometimes subtracted from to the numerator and denominator the minimum value obtained for the evaluation of an individual of the population)
- According to this method, the selection of individuals is performed by simulating n rotations of a roulette whose slices had an angle proportional to the result of the above formula, thus constituting the intended population. That is, individuals with higher adaptation value will be more likely to be chosen, however, nothing guarantees that they are, nor does it imply that the element with lower adaptation value cannot be chosen

Selection

Individual *i* has $\frac{f(i)}{\sum_i f(i)}$

Probability to be seleted

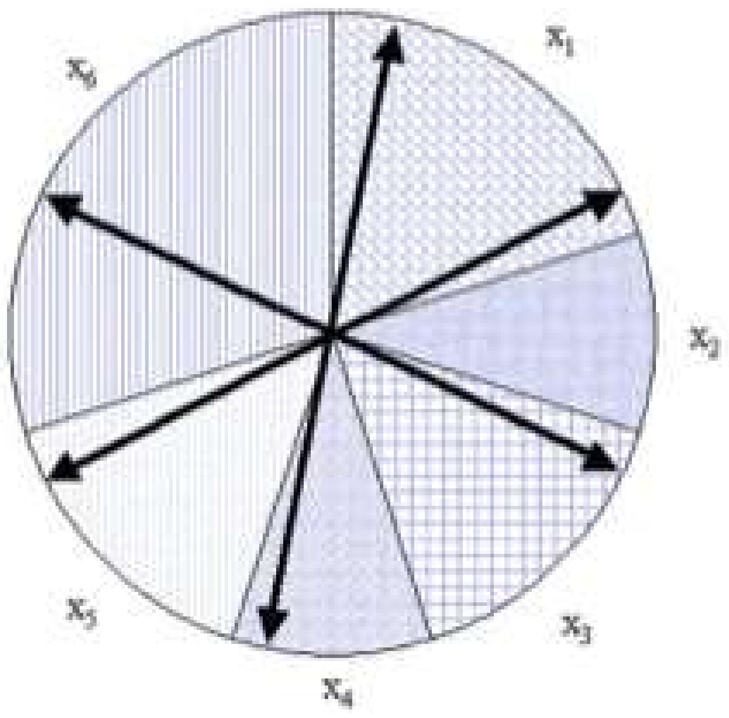


Selection

In addition to the roulette method, other proportional selection operators were also studied, such as the Universal Stochastic Selection Method

- We spin the roulette only once and have a set of equally spaced "pointers", the individuals whose slices are located on such pointers will be chosen
- Population evolution over generations tends to standardize individual adaptation values, which reduces the importance of proportional selection because it is based on the fact that there are substantial differences between individual adaptation values

Uniform Stochastic Sampling Method



Selection

- The Tournament Selection Method is a sort order selection method consisting of the organization of n tournaments between 2 or more randomly selected individuals in the population
- The selected individual is the one in the superior position in the direct confrontation between the fitness values