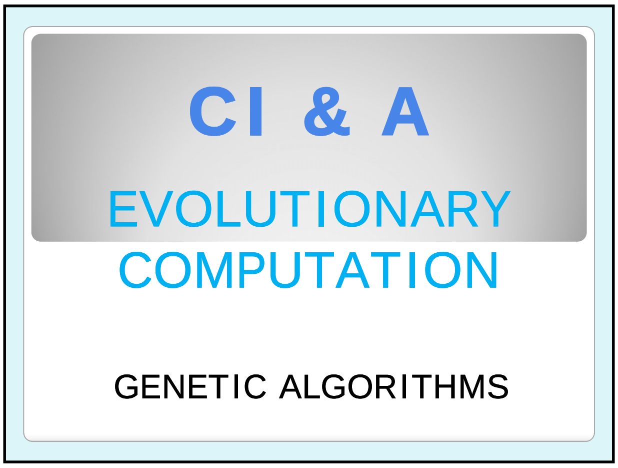
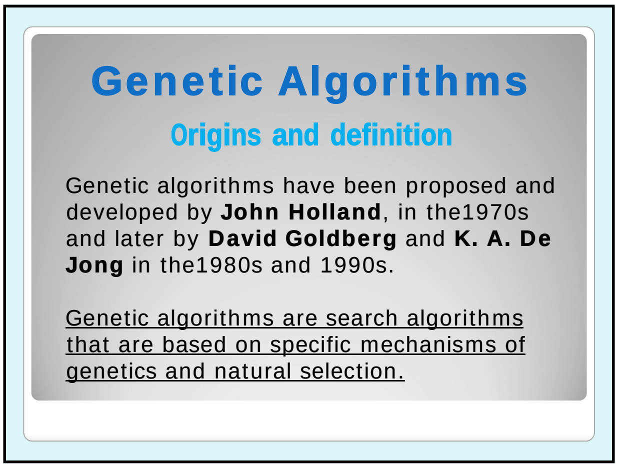


CI & A

EVOLUTIONARY COMPUTATION

GENETIC ALGORITHMS

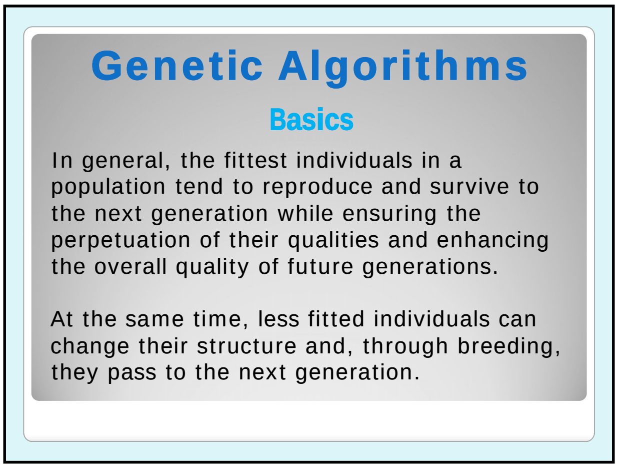


Genetic Algorithms

Origins and definition

Genetic algorithms have been proposed and developed by **John Holland**, in the1970s and later by **David Goldberg** and **K. A. De Jong** in the1980s and 1990s.

Genetic algorithms are search algorithms that are based on specific mechanisms of genetics and natural selection.

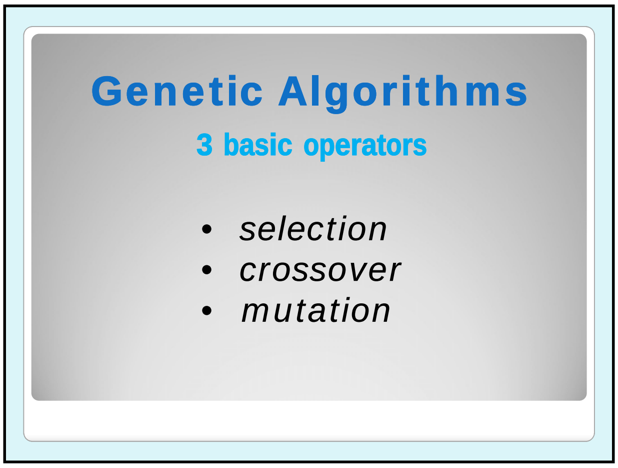


Genetic Algorithms

Basics

In general, the fittest individuals in a population tend to reproduce and survive to the next generation while ensuring the perpetuation of their qualities and enhancing the overall quality of future generations.

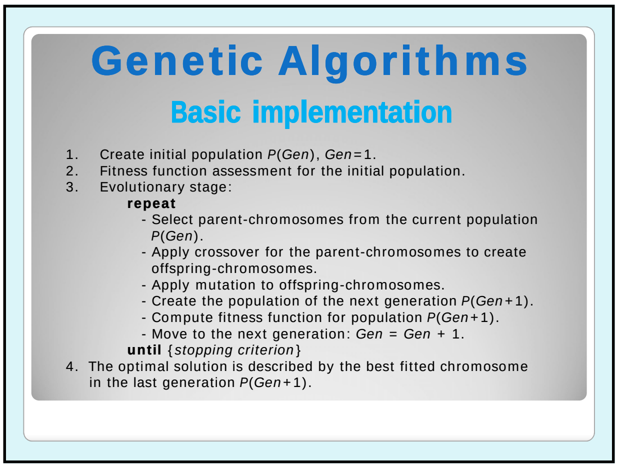
At the same time, less fitted individuals can change their structure and, through breeding, they pass to the next generation.



Genetic Algorithms

3 basic operators

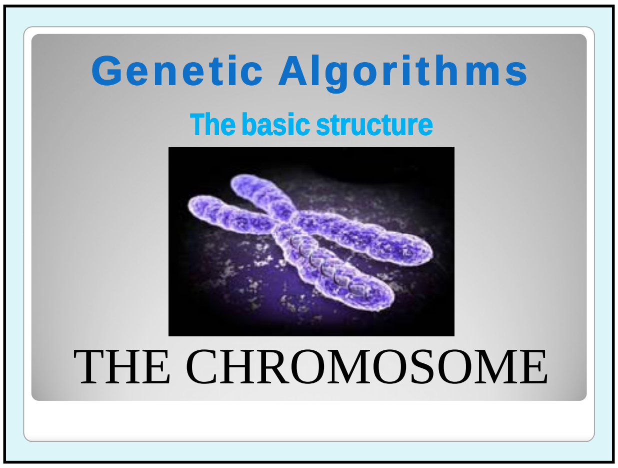
- *selection*
- *crossover*
- *mutation*



Genetic Algorithms

Basic implementation

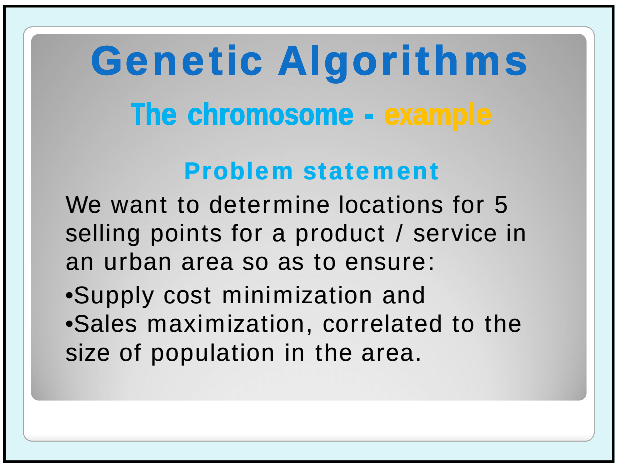
```
1.  Create initial population P(Gen), Gen=1.
2.  Fitness function assessment for the initial population.
3.  Evolutionary stage:
        repeat
          - Select parent-chromosomes from the current population
            P(Gen).
          - Apply crossover for the parent-chromosomes to create
            offspring-chromosomes.
          - Apply mutation to offspring-chromosomes.
          - Create the population of the next generation P(Gen+1).
          - Compute fitness function for population P(Gen+1).
          - Move to the next generation: Gen = Gen + 1.
        until {stopping criterion}
4. The optimal solution is described by the best fitted chromosome
   in the last generation P(Gen+1).
```



Genetic Algorithms

The basic structure

THE CHROMOSOME



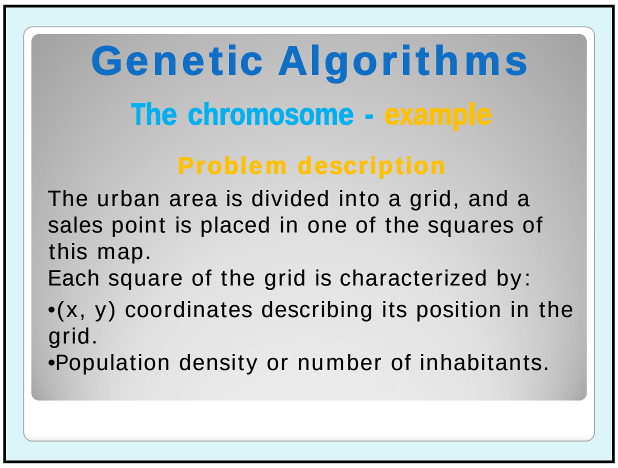
Genetic Algorithms

The chromosome - example

Problem statement

We want to determine locations for 5 selling points for a product / service in an urban area so as to ensure:

- Supply cost minimization and
- Sales maximization, correlated to the size of population in the area.



Genetic Algorithms

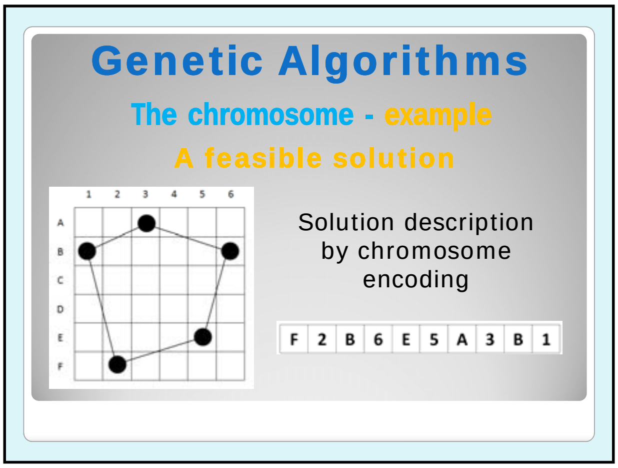
The chromosome - example

Problem description

The urban area is divided into a grid, and a sales point is placed in one of the squares of this map.

Each square of the grid is characterized by:

- (x, y) coordinates describing its position in the grid.
- Population density or number of inhabitants.



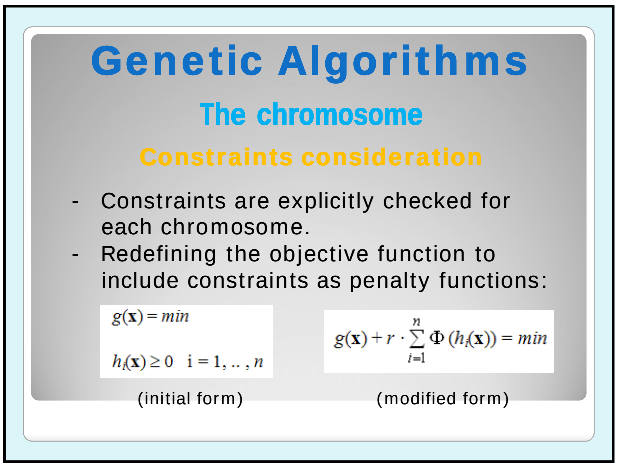
Genetic Algorithms

The chromosome - example

A feasible solution

Solution description by chromosome encoding

F	2	B	6	E	5	A	3	B	1



Genetic Algorithms

The chromosome

Constraints consideration

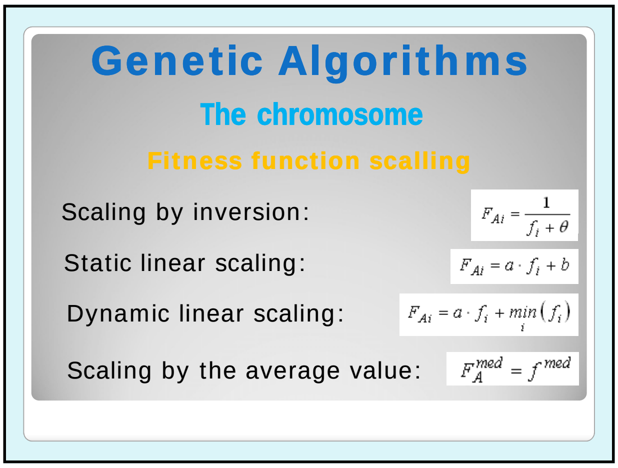
- Constraints are explicitly checked for each chromosome.
- Redefining the objective function to include constraints as penalty functions:

$$g(\mathbf{x}) = min \qquad h_i(\mathbf{x}) \ge 0 \quad \mathrm{i} = 1, .., n$$

(initial form)

$$g(\mathbf{x}) + r \cdot \sum_{i=1}^{n} \Phi\,(h_i(\mathbf{x})) = min$$

(modified form)



Genetic Algorithms

The chromosome

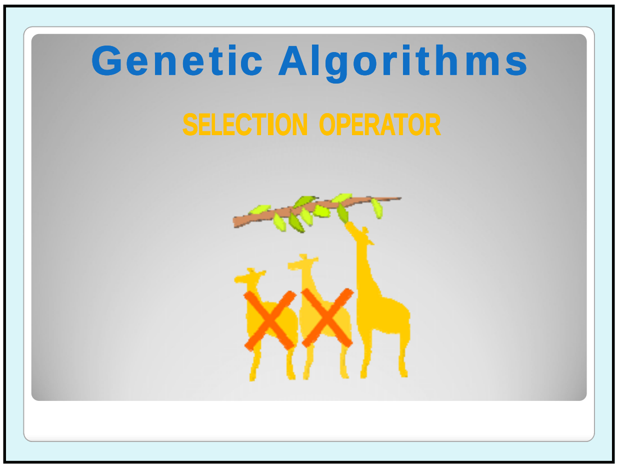
Fitness function scalling

Scaling by inversion: $F_{Ai} = \dfrac{1}{f_i + \theta}$

Static linear scaling: $F_{Ai} = a \cdot f_i + b$

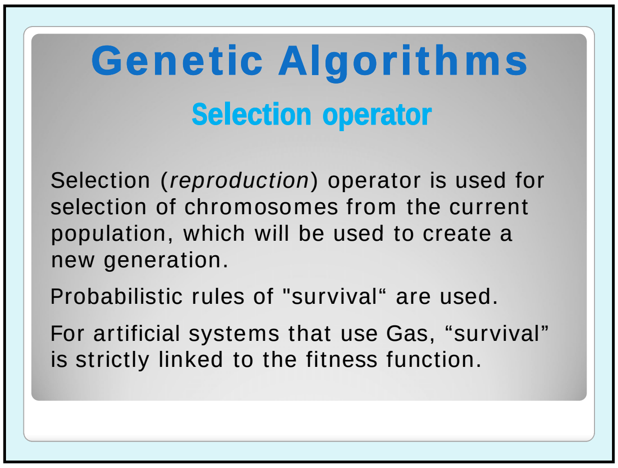
Dynamic linear scaling: $F_{Ai} = a \cdot f_i + \min_i (f_i)$

Scaling by the average value: $F_A^{med} = f^{med}$



Genetic Algorithms

SELECTION OPERATOR



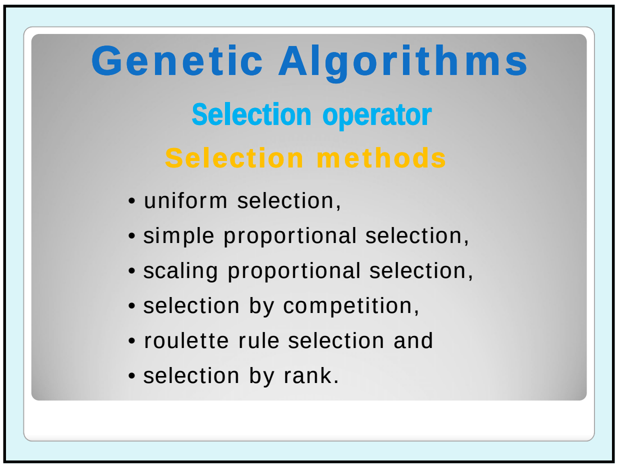
Genetic Algorithms

Selection operator

Selection (*reproduction*) operator is used for selection of chromosomes from the current population, which will be used to create a new generation.

Probabilistic rules of "survival“ are used.

For artificial systems that use Gas, “survival” is strictly linked to the fitness function.

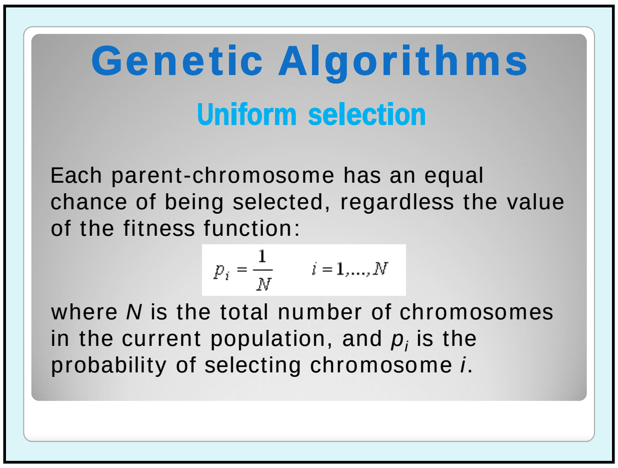


Genetic Algorithms

Selection operator

Selection methods

- uniform selection,
- simple proportional selection,
- scaling proportional selection,
- selection by competition,
- roulette rule selection and
- selection by rank.



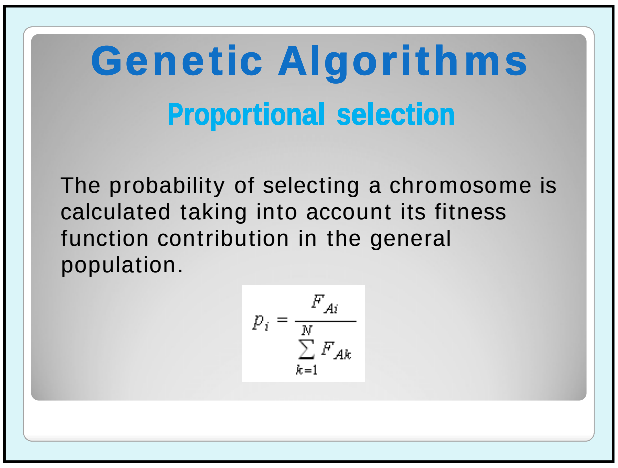
Genetic Algorithms

Uniform selection

Each parent-chromosome has an equal chance of being selected, regardless the value of the fitness function:

$$p_i = \frac{1}{N} \qquad i = 1,\ldots,N$$

where *N* is the total number of chromosomes in the current population, and p_i is the probability of selecting chromosome *i*.

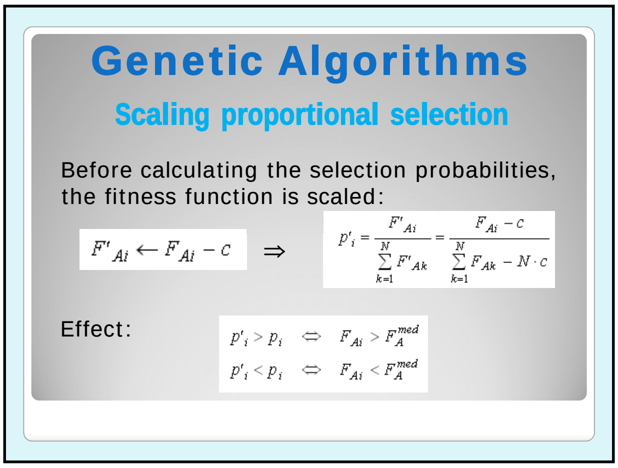


Genetic Algorithms

Proportional selection

The probability of selecting a chromosome is calculated taking into account its fitness function contribution in the general population.

$$p_i = \frac{F_{Ai}}{\sum_{k=1}^{N} F_{Ak}}$$



Genetic Algorithms

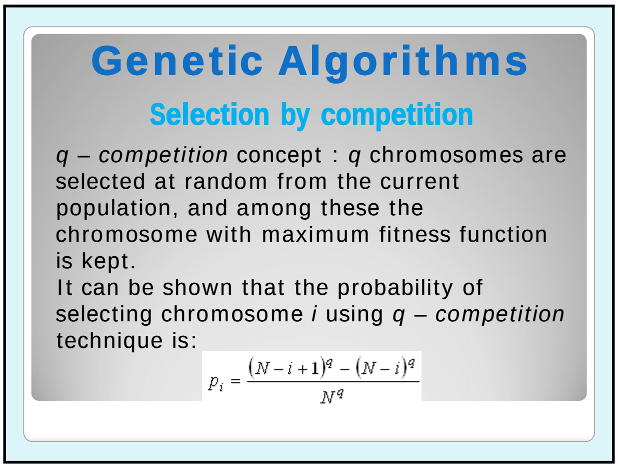
Scaling proportional selection

Before calculating the selection probabilities, the fitness function is scaled:

$$F'_{Ai} \leftarrow F_{Ai} - c \quad \Rightarrow \quad p'_i = \frac{F'_{Ai}}{\sum_{k=1}^{N} F'_{Ak}} = \frac{F_{Ai} - c}{\sum_{k=1}^{N} F_{Ak} - N \cdot c}$$

Effect:

$$\begin{aligned} p'_i > p_i &\Leftrightarrow F_{Ai} > F_A^{med} \\ p'_i < p_i &\Leftrightarrow F_{Ai} < F_A^{med} \end{aligned}$$



Genetic Algorithms

Selection by competition

q – competition concept : *q* chromosomes are selected at random from the current population, and among these the chromosome with maximum fitness function is kept.

It can be shown that the probability of selecting chromosome *i* using *q – competition* technique is:

$$p_i = \frac{(N - i + 1)^q - (N - i)^q}{N^q}$$

Genetic Algorithms

Roulette rule selection

An "animistic" implementation form for proportional selection.

The sum of the fitness function for all chromosomes of the current population is associated with the entire length of roulette, which is then divided into sectors with length proportional to the fitness function of each chromosome.

Genetic Algorithms

Roulette rule selection

Example: 5 chromosomes, with fitness functions:

①50 ② 350 ③ 150 ④ 250 ⑤ 200

Circumference of the roulette :

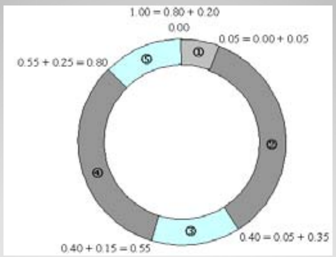
50 + 350 + 150 + 250 + 200 = 1000

Selection probabilities:

0.05 0.35 0.15 0.25 0.20.

Genetic Algorithms

Roulette rule selection



Genetic Algorithms

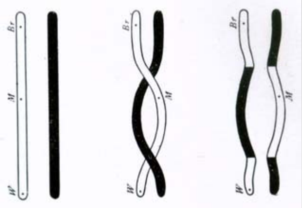
Selection by rank

Selection by rank need the assessment of fitness functions for all chromosomes and the rearrangement of chromosomes in descending order: the most fitted chromosome has rank 1 and the weakest chromosome has rank *N*. Further, selection probabilities are calculated based only on the rank of each chromosome, without using fitness functions:

$$p_i = \frac{1}{N}\left[\eta - 2\cdot(\eta - 1)\cdot\frac{i-1}{N-1}\right]$$

Genetic Algorithms

CROSSOVER OPERATOR



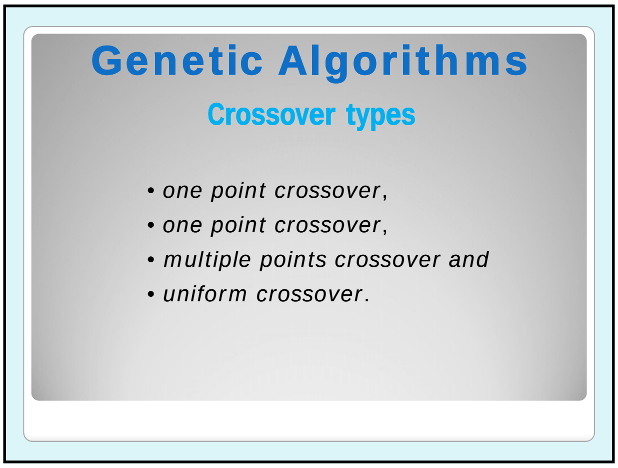
Genetic Algorithms

Crossover operator

Two *parent-chromosomes* are chosen to recombine and give birth to two new *offspring-chromosomes*, which pass into the next generation.

The crossover of parent-chromosomes occurs with probability p_l, so it is possible that the two parent-chromosomes to pass to the next generation without modification.

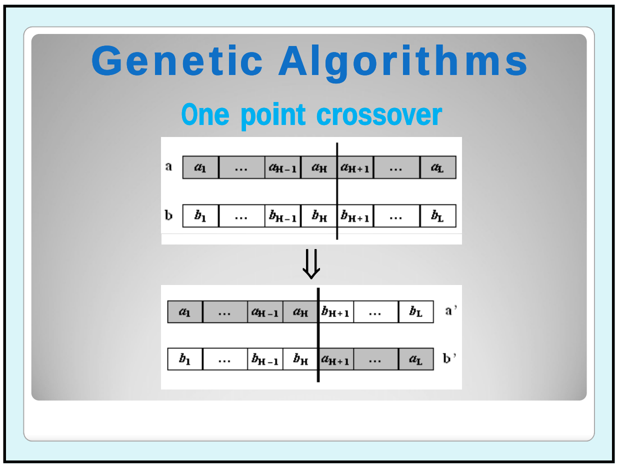
Probability p_l: values between 0.6 and 0.95.



Genetic Algorithms

Crossover types

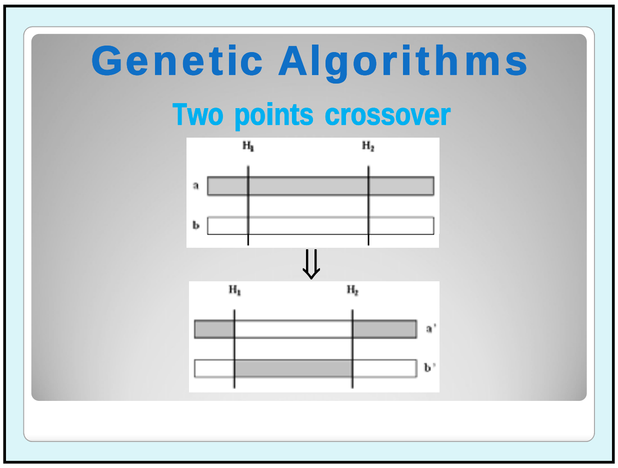
- *one point crossover,*
- *one point crossover,*
- *multiple points crossover and*
- *uniform crossover.*



Genetic Algorithms

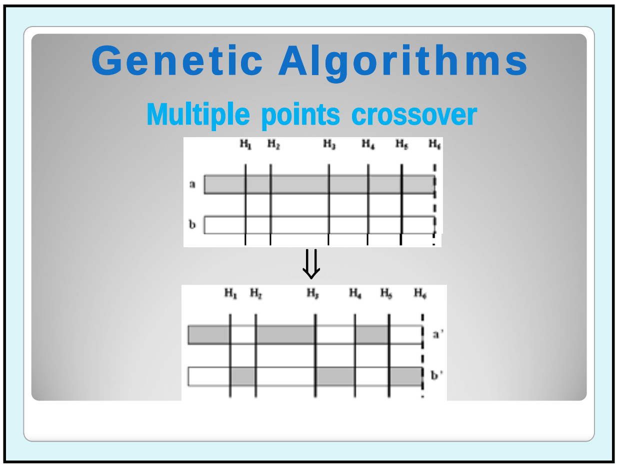
One point crossover

a
a_1 … a_{H-1} a_H a_{H+1} … a_L
b
b_1 … b_{H-1} b_H b_{H+1} … b_L
⇓
a_1 … a_{H-1} a_H b_{H+1} … b_L
a'
b_1 … b_{H-1} b_H a_{H+1} … a_L
b'



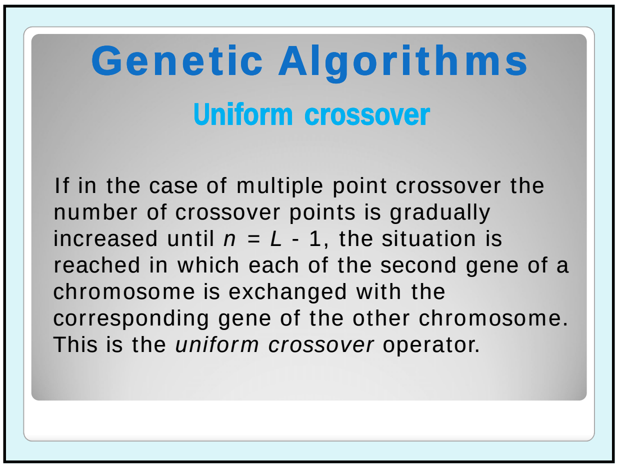
Genetic Algorithms

Two points crossover



Genetic Algorithms

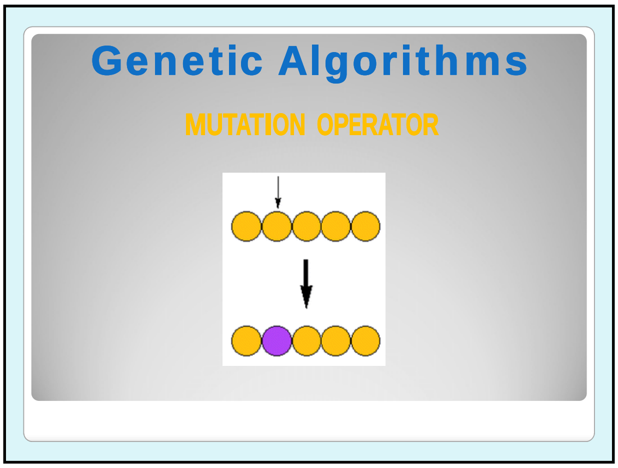
Multiple points crossover



Genetic Algorithms

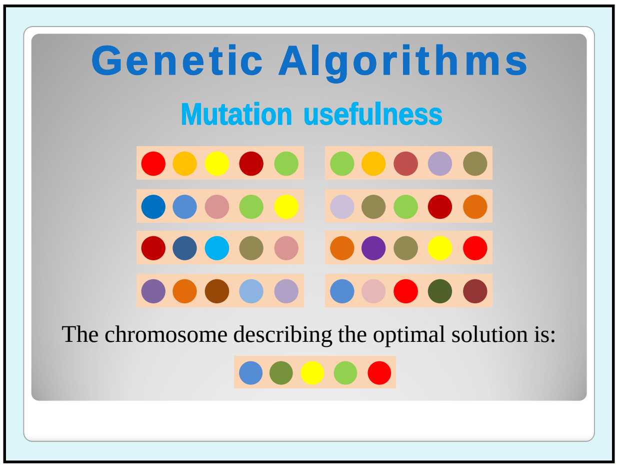
Uniform crossover

If in the case of multiple point crossover the number of crossover points is gradually increased until n = L - 1, the situation is reached in which each of the second gene of a chromosome is exchanged with the corresponding gene of the other chromosome. This is the *uniform crossover* operator.



Genetic Algorithms

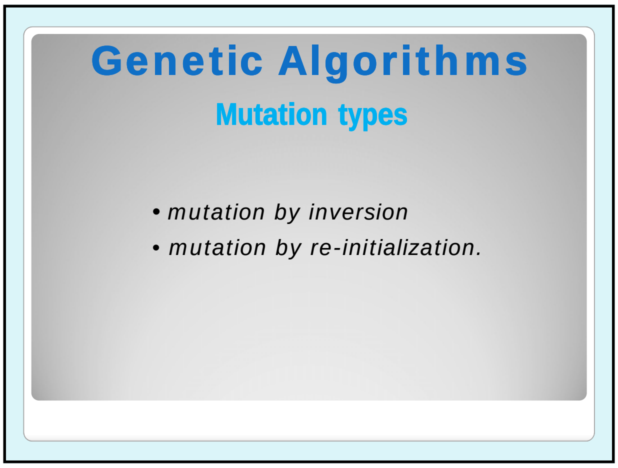
MUTATION OPERATOR



Genetic Algorithms

Mutation usefulness

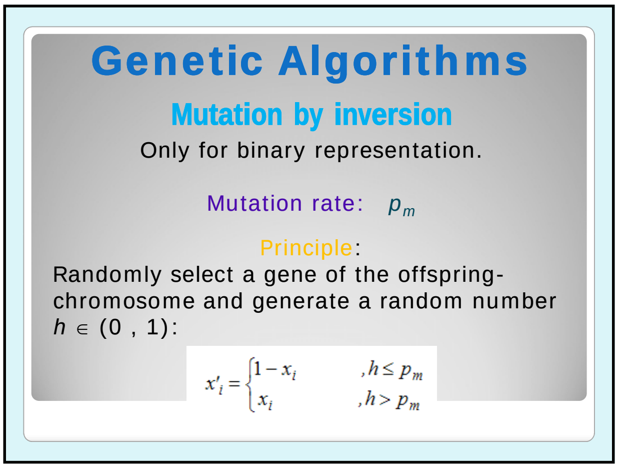
The chromosome describing the optimal solution is:



Genetic Algorithms

Mutation types

- *mutation by inversion*
- *mutation by re-initialization.*



Genetic Algorithms

Mutation by inversion

Only for binary representation.

Mutation rate: p_m

Principle:

Randomly select a gene of the offspring-chromosome and generate a random number $h \in (0 , 1)$:

$$x'_i = \begin{cases} 1 - x_i & , h \le p_m \\ x_i & , h > p_m \end{cases}$$

Genetic Algorithms

Mutation by re-initialization

Only for numerical representations (integer or real).

Mutation rate: p_m

Types:

- basic mutation by re-initialization $[\ a_i\ ,\ b_i\ ]$
- mutation with fixed step; $x'_i = x_i \pm \delta$
- uniform mutation $[\delta_{\min}\ ,\ \delta_{\max}\ ]$
- others

Genetic Algorithms

GENERATION OF NEW POPULATION

Genetic Algorithms

Replacing procedures

- *complete replacement*
- *selective replacement*

ELITISM