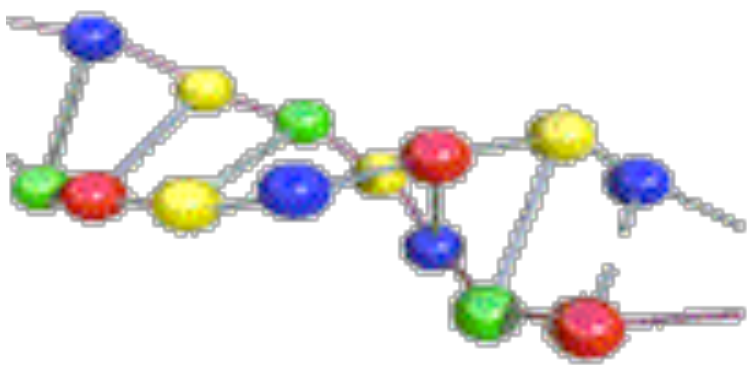
Genetic Algorithms (GAs)



Developed by John Holland, 1975

Holland, John. 1975 "*Adaption in Natural and Artificial Systems*"

www.obitko.com/tutorials/genetic-algorithms/index.php

A **genetic algorithm** is inspired by Darwin's theory of evolution – by the biological process of evolution

Problems tackled by a genetic algorithm have solutions that evolve

i.e., computationally, by a **simulated evolutionary process**

Has utilization for optimization and machine learning

Features:

- **Stochastic**: different results from different runs
- Maintain a population of solutions at each time
- Solutions are encoded as **chromosomes**
- **Crossover** and **mutation** operate during the reproduction.
- **Survival of the fittest**

Genetic Algorithms (GAs)

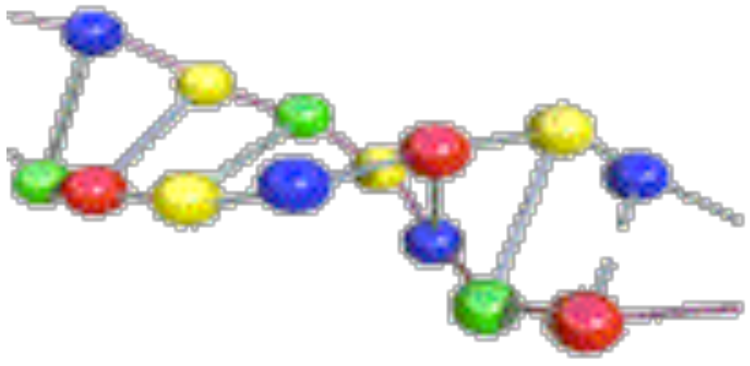
Problems tackled by a genetic algorithm have solutions that evolve
i.e., computationally, it is **simulated evolutionary process**

The process starts with a given **population** expressed as a set of **solutions** (represented by **chromosomes**).

At each stage in the process, solutions from one populations are taken to form a new population. This is motivated by a hope, that the new population will be better than the old one.

Solutions which are selected to reproduce new solutions (**offsprings**) are selected according to their **fitness** - the more suitable they are the more chances they have to reproduce.

This process is repeated until some condition is satisfied; for example, number of populations or improvement of the best solution.



GA-Reproduction Rules

Crossover (Recombination)

During reproduction, **genes** from parents are recombined to form in some way the whole new chromosome.

Mutation

Mutation changes the elements of DNA of the newly created offspring.

Fitness evaluation

Fitness function is used to evaluate the success of the new offspring.

GA-the basic algorithm

[Start] Generate random population of n chromosomes (suitable solutions for the problem)

[New population] Create a new population by repeating following steps until the new population is complete

- **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
- **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
- **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
- **[Fitness]** Evaluate the fitness f(x) of each chromosome x in the population
- **[Accepting]** Place new offspring in a new population

Loop

[Test] If the end condition is satisfied, stop, and return the best solution in current population

[Replace] Use new generated population for a further run of algorithm

Chromosome representation

Encoding a chromosome by **bit** strings within a given domain.

The chromosome should in some way contain information about solution which it represents.

Each chromosome has one binary string.

Each bit in this string can represent some characteristic of the solution. For example, the whole string can represent a number (ex. 0100 = 4).

Chromosome 01	1101 1001 0011 1011
Chromosome 02	1011 1110 0001 1001

Cross-over

Crossover selects genes from parent chromosomes and creates a new offspring. The simplest way how to do this is to choose randomly a **crossover point** and everything before this point point copy from a first parent and then everything after a crossover point copy from the second parent.

Crossover Point

Chromosome 01	1101 1001 0011 1011
Chromosome 02	1011 1110 0001 1001

Offspring 01	1101 1001 0001 1001
Offspring 02	1011 1110 0011 1011

Mutation

Usually, **mutation** takes place after a ***crossover*** has been performed,.

This is to prevent falling all solutions in population into a **local optimum** of solved problem.

Mutation changes randomly the new offspring.

For example, in a binary-encoded presentation,

we can switch a few randomly chosen bits from 1 to 0, or vice versa.

Mutation Maps

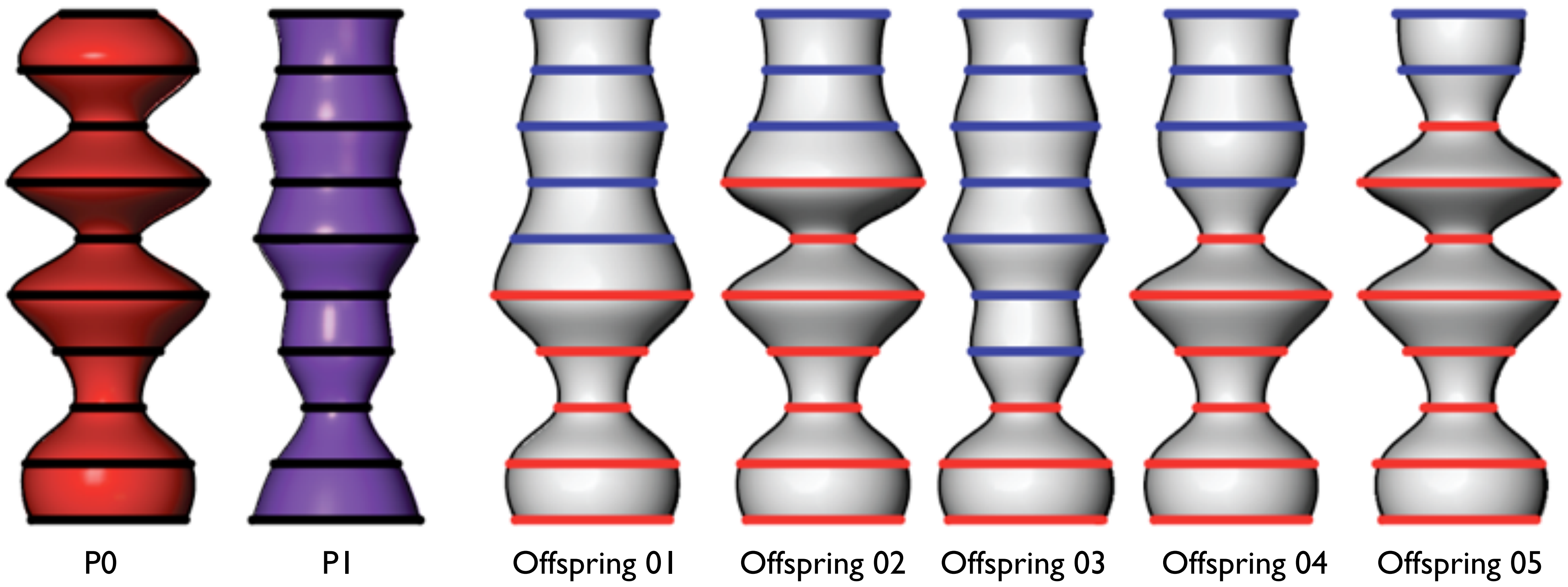
Offspring 01	1101 1001 0001 1001
Offspring 02	1011 1110 0011 1011

Mutation Maps

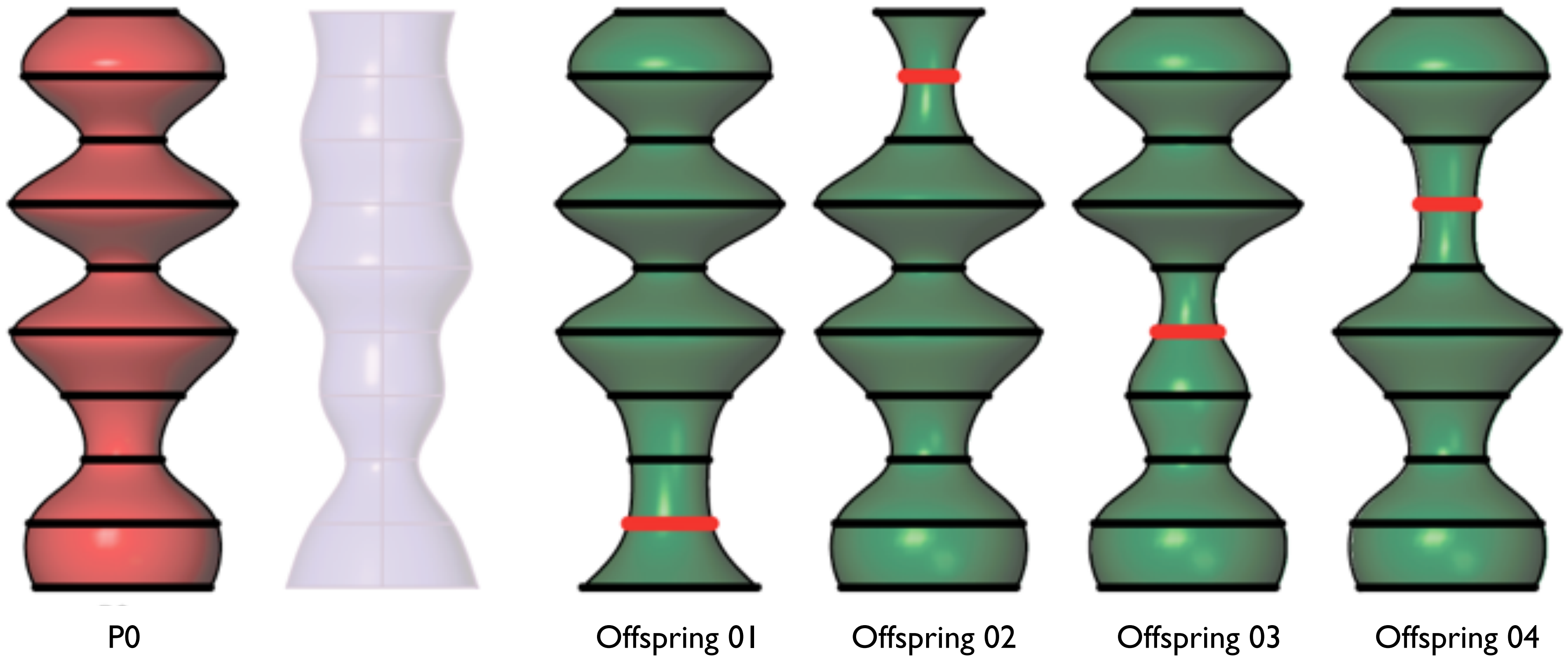
after mutation:

Mutated Offspring 01	1101 **0**001 0**1**01 1001
Mutated Offspring 02	1011 1110 0**1**11 101**0**

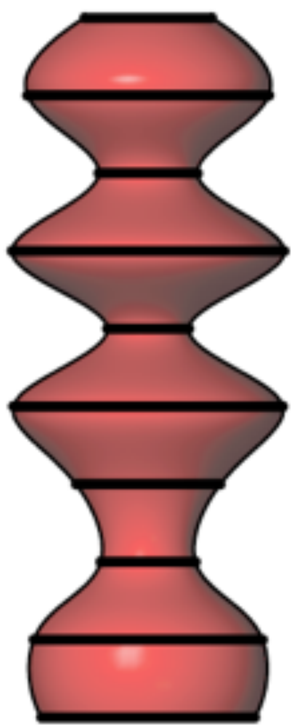
Examples: Crossover



Examples: Mutation



Chromosome Encoding - an array of real numbers



P0

$radius = min + t\,(max - min) \qquad 0 \le t \le 1$

number represents the normalized radius of each section circle

Chromosome P0	0.75	0.82	0.34	0.51	0.96	0.29	0.97	0.35	0.86	0.45

Encoding scheme

double representation

Chromosome P0	0.75	0.82	0.34	0.51	0.96	0.29	0.97	0.35	0.86	0.45

||

binary string representation

Chromosome P0	11000011	11010001	01010111	10000100	11110110	01001011	11111000	1011010	11011100	01110100

conversion from binary string to integer

11111111	255
00000000	0

0.75 * 255 ~= 195

$195 = 128+64+2+1 = 2^7+2^6+2^1+2^0 = 11000011$