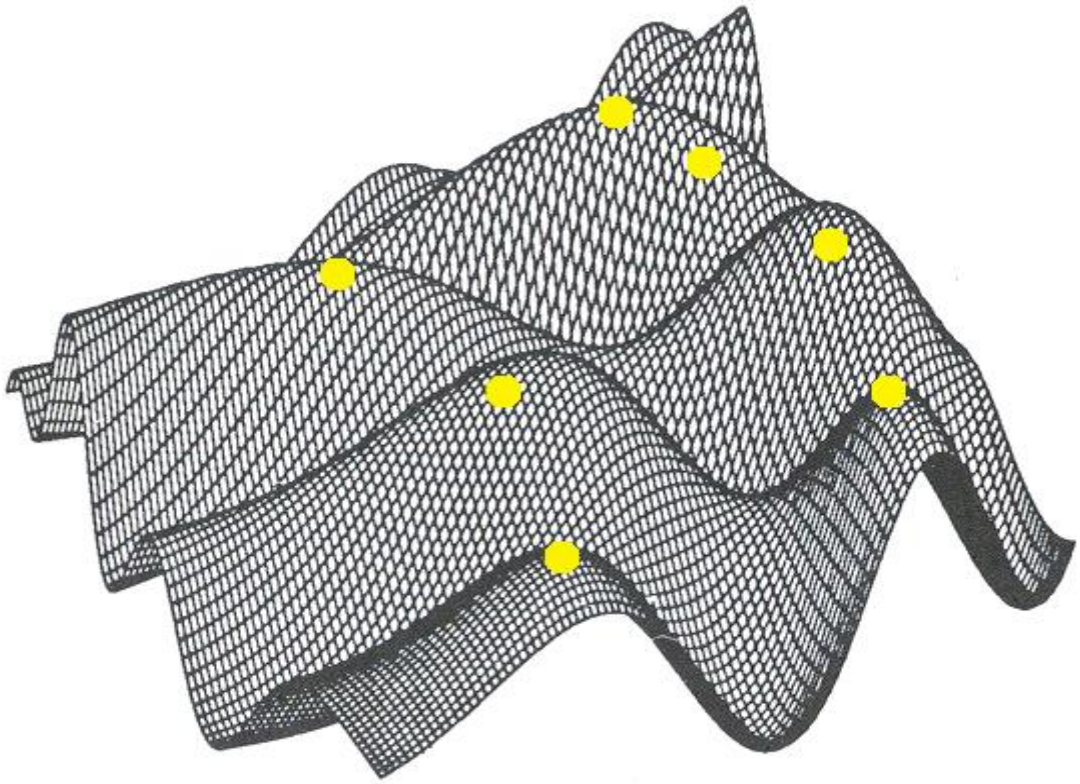
Experimental Techniques in Physics Supported with AI/ML:

Genetic Algorithms

Lecture 8, summer 2023/2024

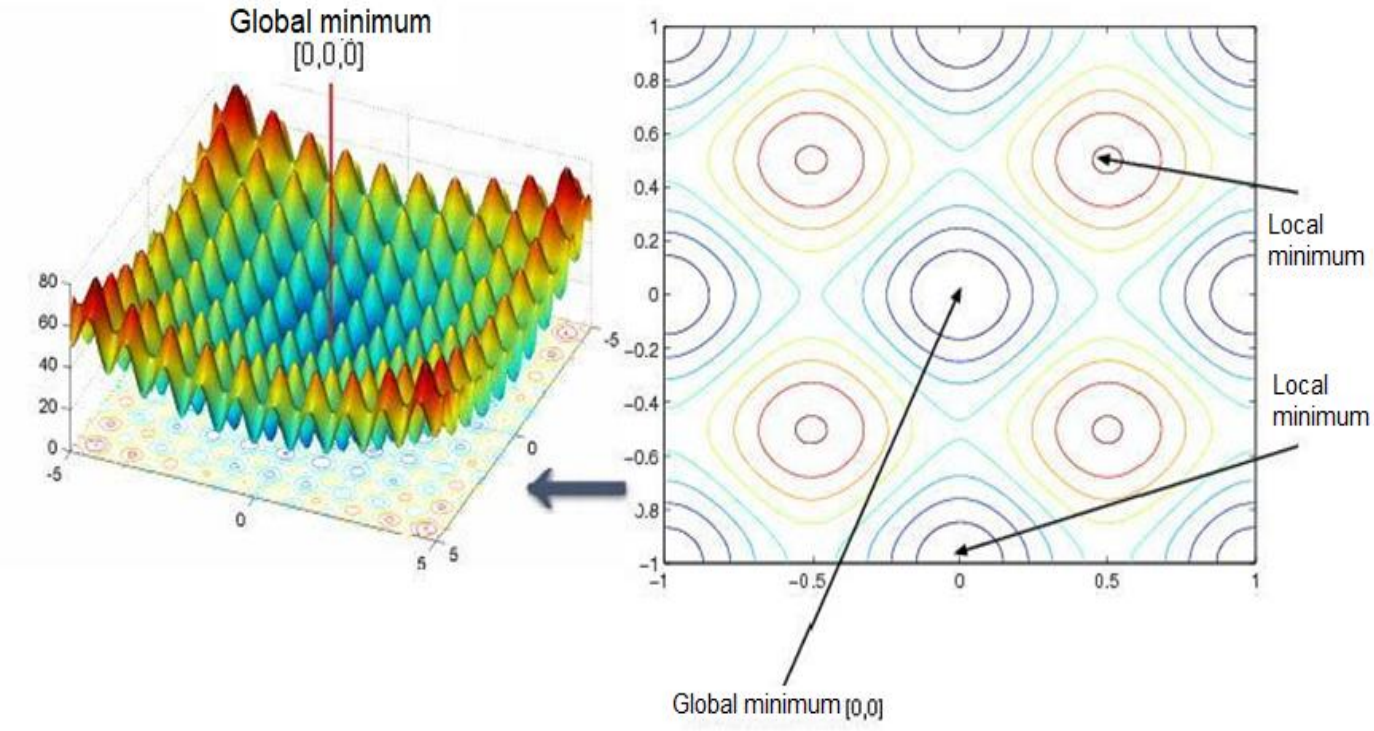
Optimization issues

The sources of problems in directed analytical optimization are mainly local extremes.



Optimization issues

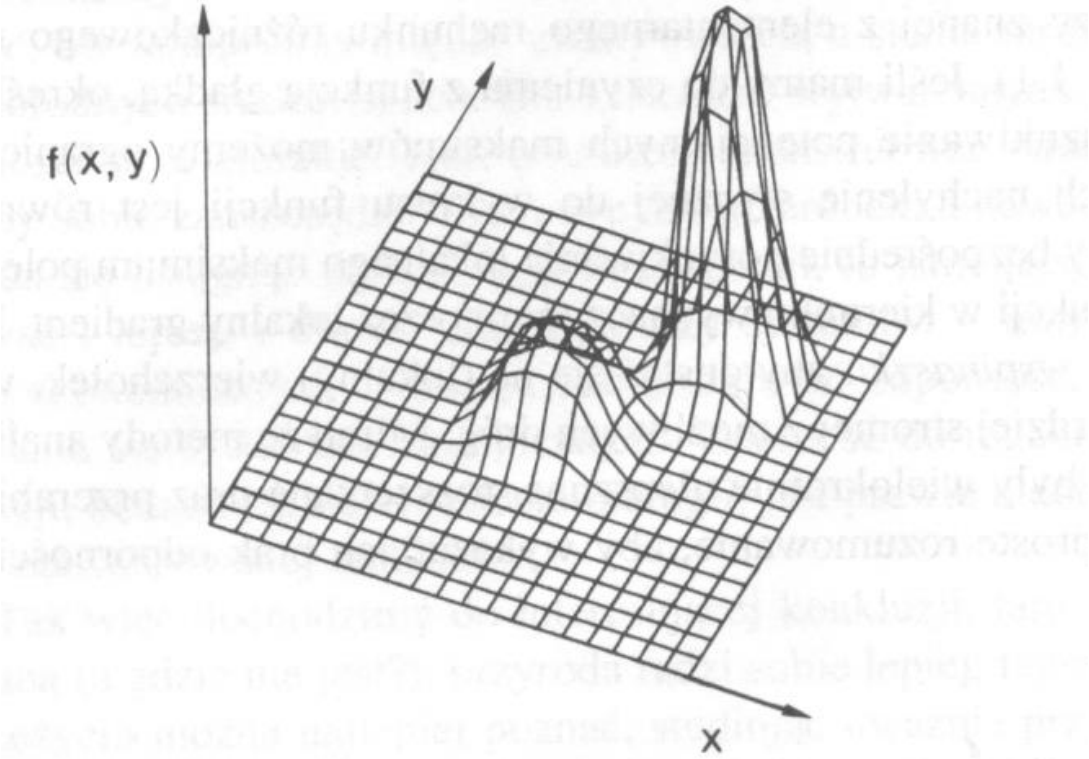
Here you can see how difficult it can be to find the global minimum.



The graph of the so-called Rastrigin function shown in the figure illustrates the difficulties that can be encountered when searching for an optimum. This function has a minimum value at the point (0,0,0), but before the search algorithm finds this global minimum, it may encounter many local minima

Optimization issues

Enumerative methods - search the entire space (element by element) - disadvantage: time-consuming algorithm.

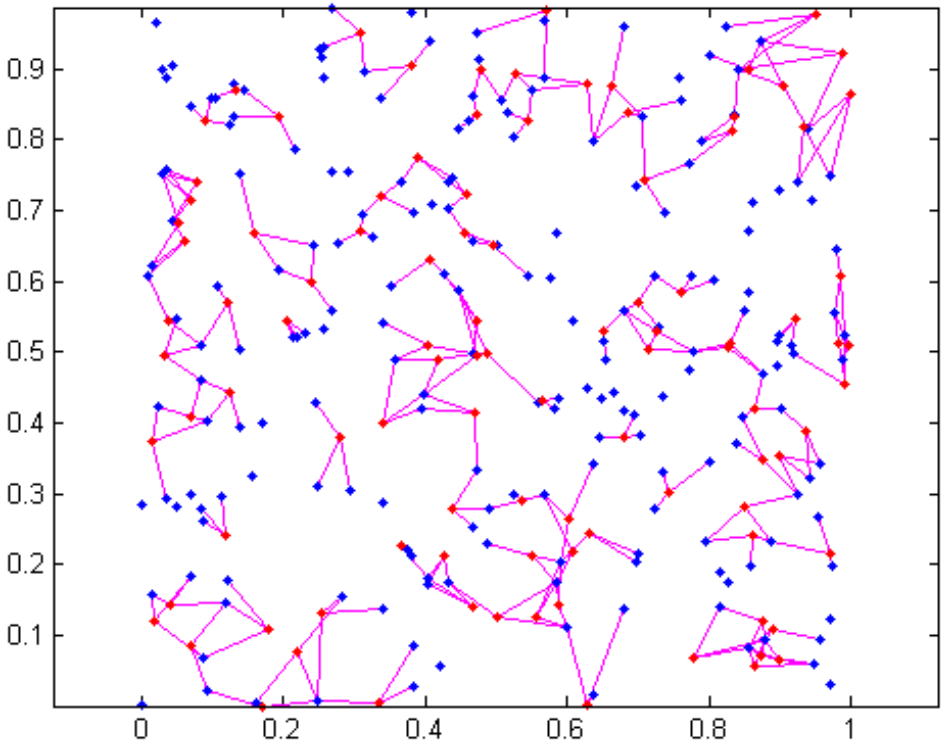


Optimization issues

Probabilistic optimization methods are free from the problem of local minima.

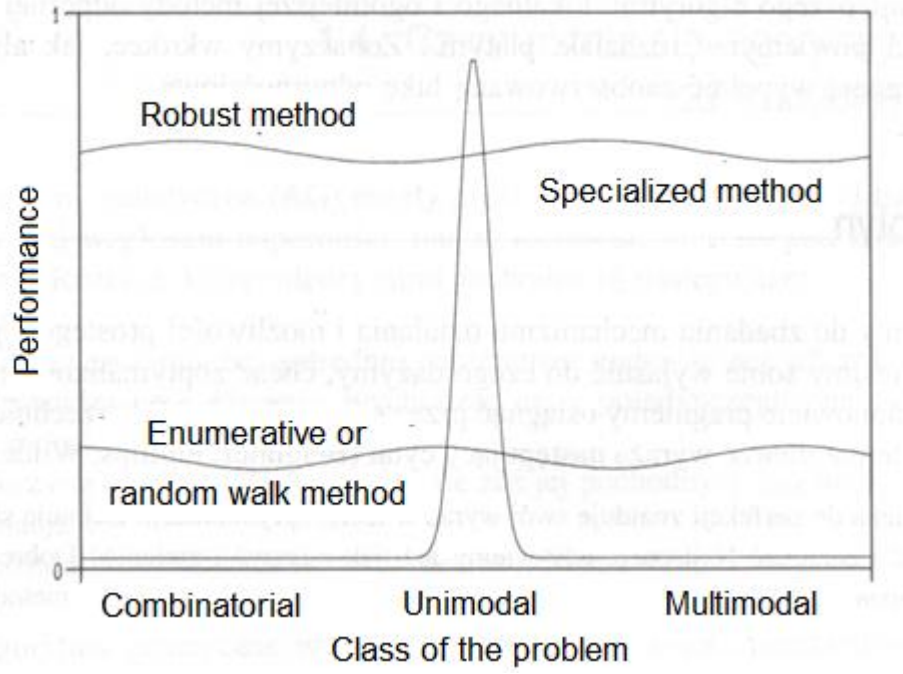
Optimization issues

Stochastic search for solutions does not guarantee success.



Optimization issues

Comparison of the efficiency (performance) of the general-purpose method (general-purpose algorithm), which consistently maintains efficiency near the average efficiency (average), and the specialized method (highly specialized algorithm), which takes into account the specific characteristics of the task.



Genetic algorithms

Motto:

Instead of laboriously searching for the best solution to an IT problem, it is better to let the computer grow the solution on its own!

Genetic algorithms

Genetic algorithms are mainly used to solve optimization tasks

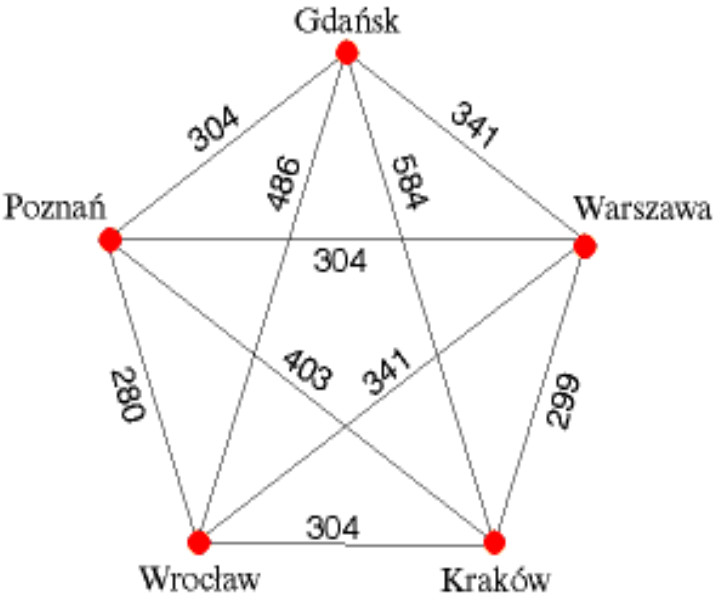
Genetic algorithms

Many optimization problems are characterized by the fact that finding an exact solution can take a very long time.

Example: The travelling salesman problem

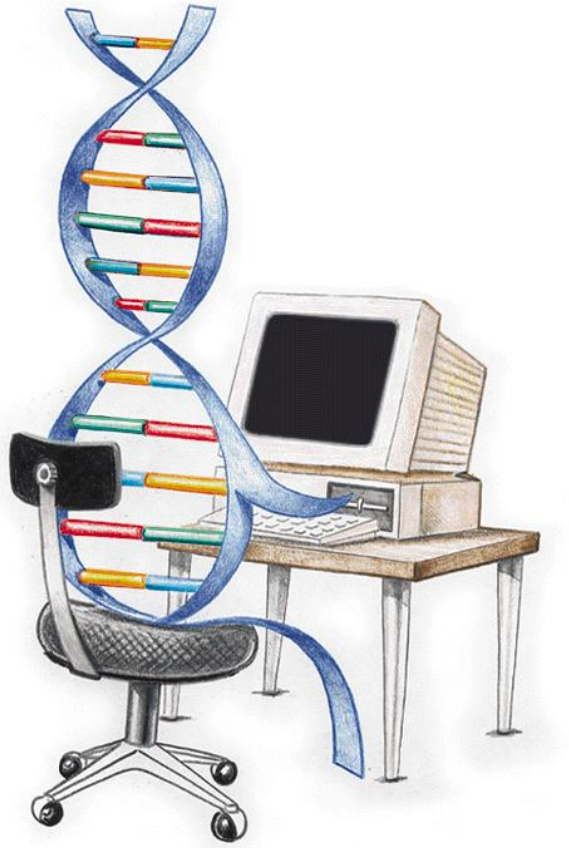
The time it takes to solve the travelling salesman problem depending on the number of cities (assuming that the computer processes one million instructions per second)

For comparison - the number of microseconds since the Big Bang in which our Universe was born is on the order of 10^{24}.



Number of cities	10	50	100	300
Time [microseconds]	~$3.6*10^6$	~10^{16}	~10^{31}	~10^{623}

Evolutionary computing - genetic algorithms



The search for the optimal solution (obtained by computer) is guided by the process of evolution.

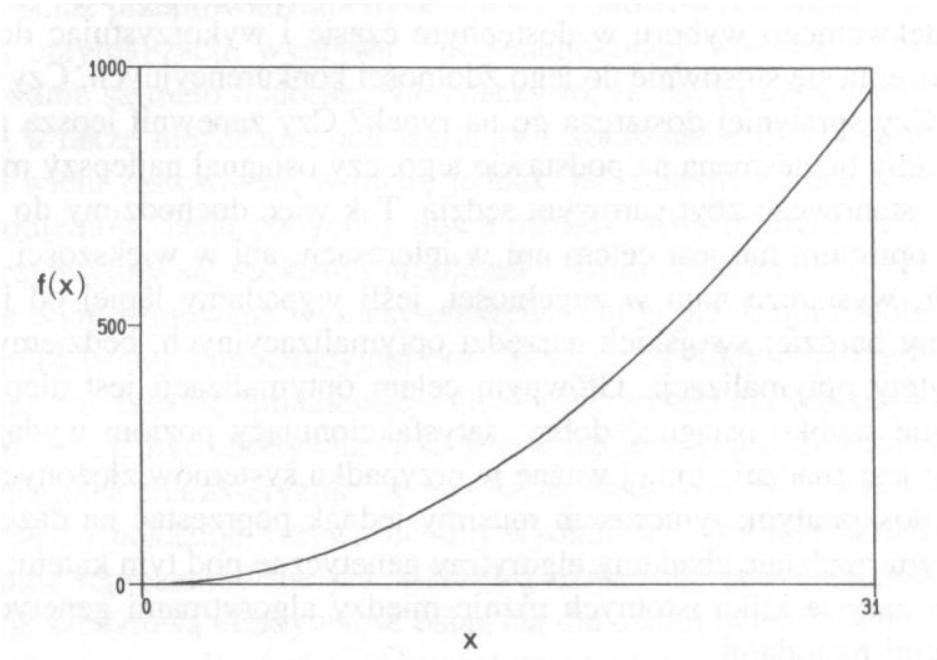
Genetic algorithms

In order for genetic algorithms (AGs) to outperform their more traditional relatives in terms of robustness, they must differ from them in several key respects:

1. GAs do not process task parameters directly, but rather their encoded form.
2. GAs conduct searches starting not from a single point, but from a certain population of them.
3. GAs use only the objective function, not its derivatives or other auxiliary information.
4. GAs use probabilistic rather than deterministic selection rules.

Genetic algorithms: maximization of the function $f(x) = x^2$

Genetic algorithms require encoding the set of parameters of an optimization task in the form of a finite sequence of characters from some finite alphabet.

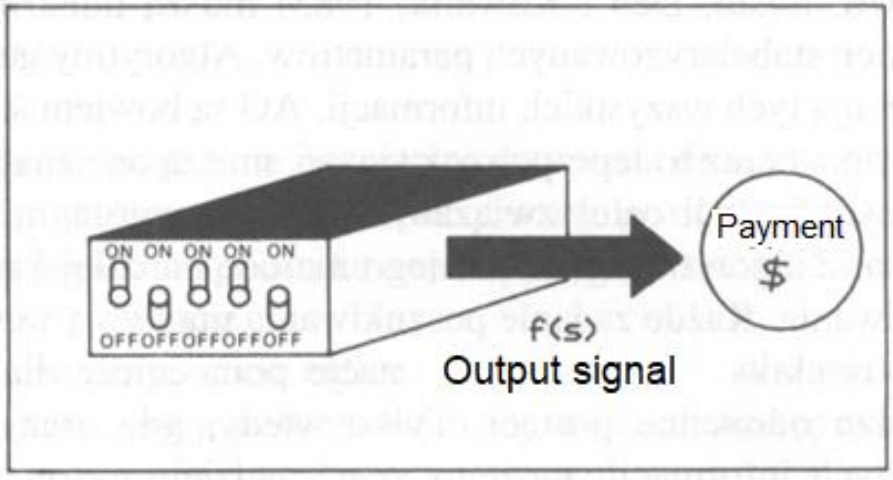


Genetic algorithms: maximization of the function $f(x) = x^2$

Coding:

Consider the problem of setting the switches in a "black box."

5 switches on the input $\rightarrow$ output signal $f = f(s)$, whre s describes a particular switch setting.



Genetic algorithms

•In many optimization methods, our actions involve carefully moving from vertex to vertex in some decision space, according to a selection rule that determines the next vertex.

•This is not a safe way, as it is a great recipe for locating false (i.e., non-global) maxima in multimodal (containing many vertices) search spaces.

Genetic algorithms

Genetic algorithms work on a rich "base" of vertices (i.e., a population of code strings) at the same time, climbing many vertices simultaneously. As a result, the chance of getting stuck on a local vertex is lower than with point-by-point methods.

Genetic algorithms: maximization of the function $f(x) = x^2$

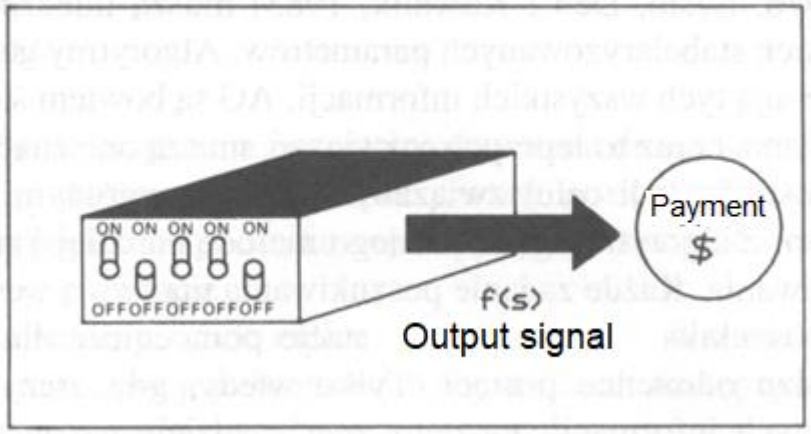
The genetic algorithm starts with an initial generation of code strings and then generates successive populations of strings. In our example, we could randomly, by making successive flips of a coin (eagle = 1, heads = 0), obtain, for example, the following initial population of size $n = 4$ (small by the standards adopted for genetic algorithms):

0 1 1 0 1

1 1 0 0 0

0 1 0 0 0

1 0 0 1 1



Genetic algorithms

- In order to conduct an efficient search for more and better solutions, genetic algorithms need to know only the magnitudes of the payoffs (i.e., the values of the objective function) associated with each code sequence.

- Genetic algorithms determine the direction of the search using probabilistic selection rules. However, this is directed randomness - toward areas where significant improvements in performance can be expected.

Genetic algorithms

The robustness of genetic algorithms is determined by four factors:

- parameter coding,
- operating on populations,
- use of minimal information about the task,
- randomized operations.

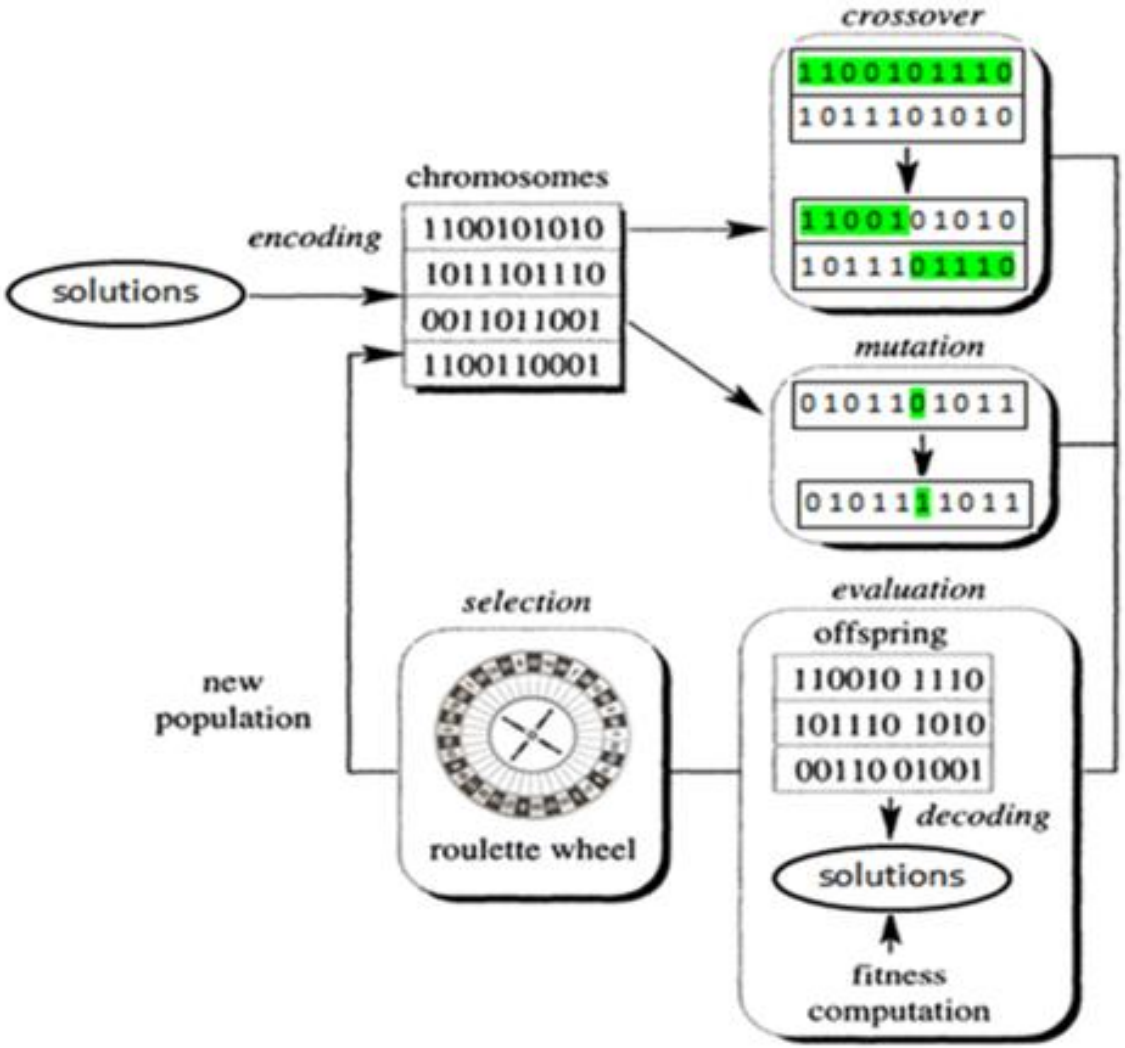
The mechanism of operation of an elementary genetic algorithm involves copying strings and replacing subsequences.

Genetic algorithms

The elementary genetic algorithm is constructed from the following three operations:

1) reproduction,
2) crossover,
3) mutation.

The scheme of genetic algorithm



Genetic algorithm: reproduction

Reproduction is the process by which individual code strings are duplicated in a ratio that depends on the values that the objective function *f* (biologists call it the adaptation function) assumes for them. Intuitively, the function *f* represents some measure of profit, utility or some other quantity that we would like to maximize.

Adaptation-dependent reproduction is the idea that code strings with higher adaptation have a higher probability of introducing one or more descendants into the next generation. This is a substitute for Darwin's principle of **natural selection**.

Genetic algorithm: reproduction

The reproduction operation can be implemented algorithmically in many ways: e.g. roulette method, tournament method, etc. Perhaps the simplest way is to simulate a properly calibrated rotating disc (roulette wheel), where each code string from the population corresponds to a sector of size proportional to the adaptation. Continuing with the considered example of four code strings for functions $f(x) = x^2$:

0 1 1 0 1

1 1 0 0 0

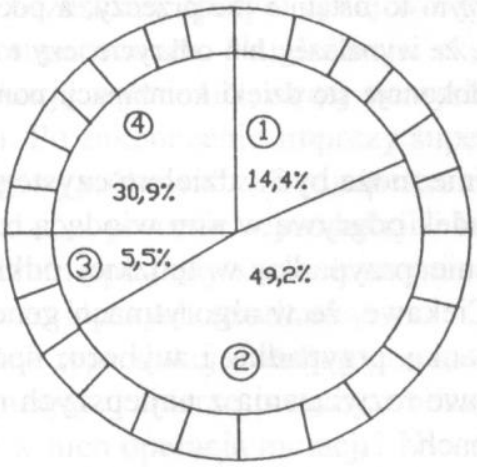
0 1 0 0 0

1 0 0 1 1

Genetic algorithm: reproduction

Table 1 Code strings and their adaptation indices in the exemplary task.

No	Code string	Goodness of fit	% Total
1	01101	169	14,4
2	11000	576	49,2
3	01000	64	5,5
4	10011	361	30,9
Total		1170	100



Each time a new descendant is needed, we run the roulette by selecting one of the candidates for reproduction. In this way, better adapted code strings introduce more descendants into the next generation. For each string selected, we create an exact replica. It is included in the new intermediate generation, in the so-called *parent pool*.

Genetic algorithm: crossover

The crossover operation proceeds in two stages.

In the first, we randomly associate strings from the parent pool into pairs.

Then each pair undergoes the crossover process. We randomly select (with equal probability) one of the positions (*crossover point k*) from among *l*-1 initial positions in the code string (where *l* - the length of the string), after which we swap all characters from position k+1 to *l* inclusive in both elements of the pair, thus creating two new strings.

Genetic algorithm: crossover

The crossover operation takes place in two stages.

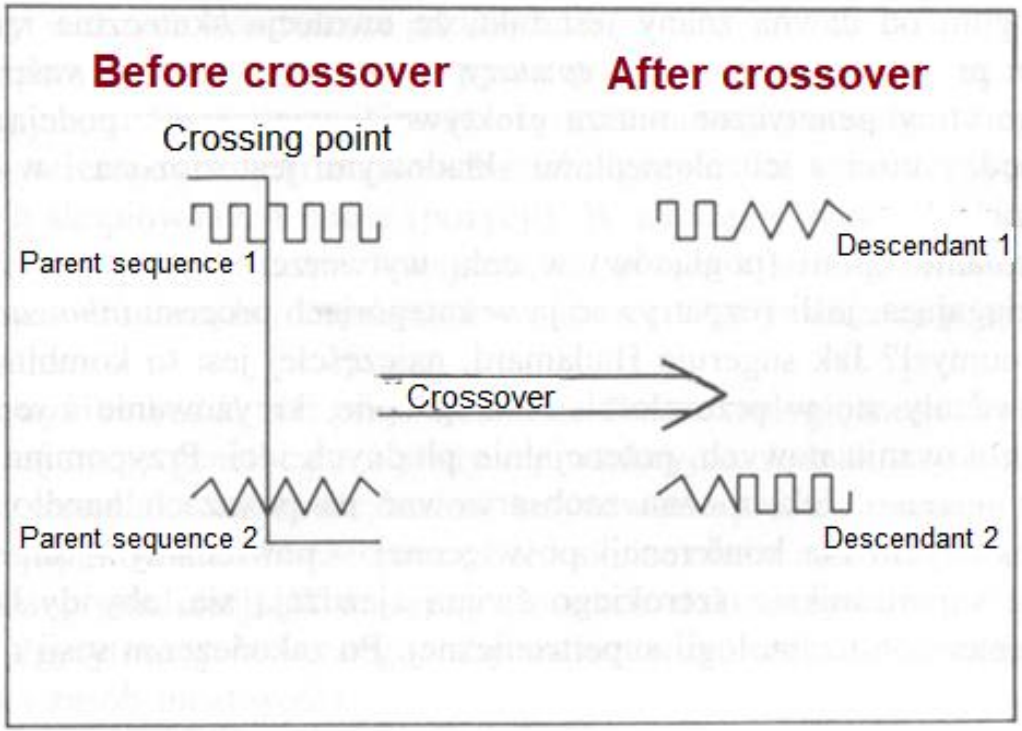
A1 = 0 1 1 0 | 1

A2 = 1 1 0 0 | 0

k = 4 (crossover point)

A1’ = 0 1 1 0 0

A2’ = 1 1 0 0 1



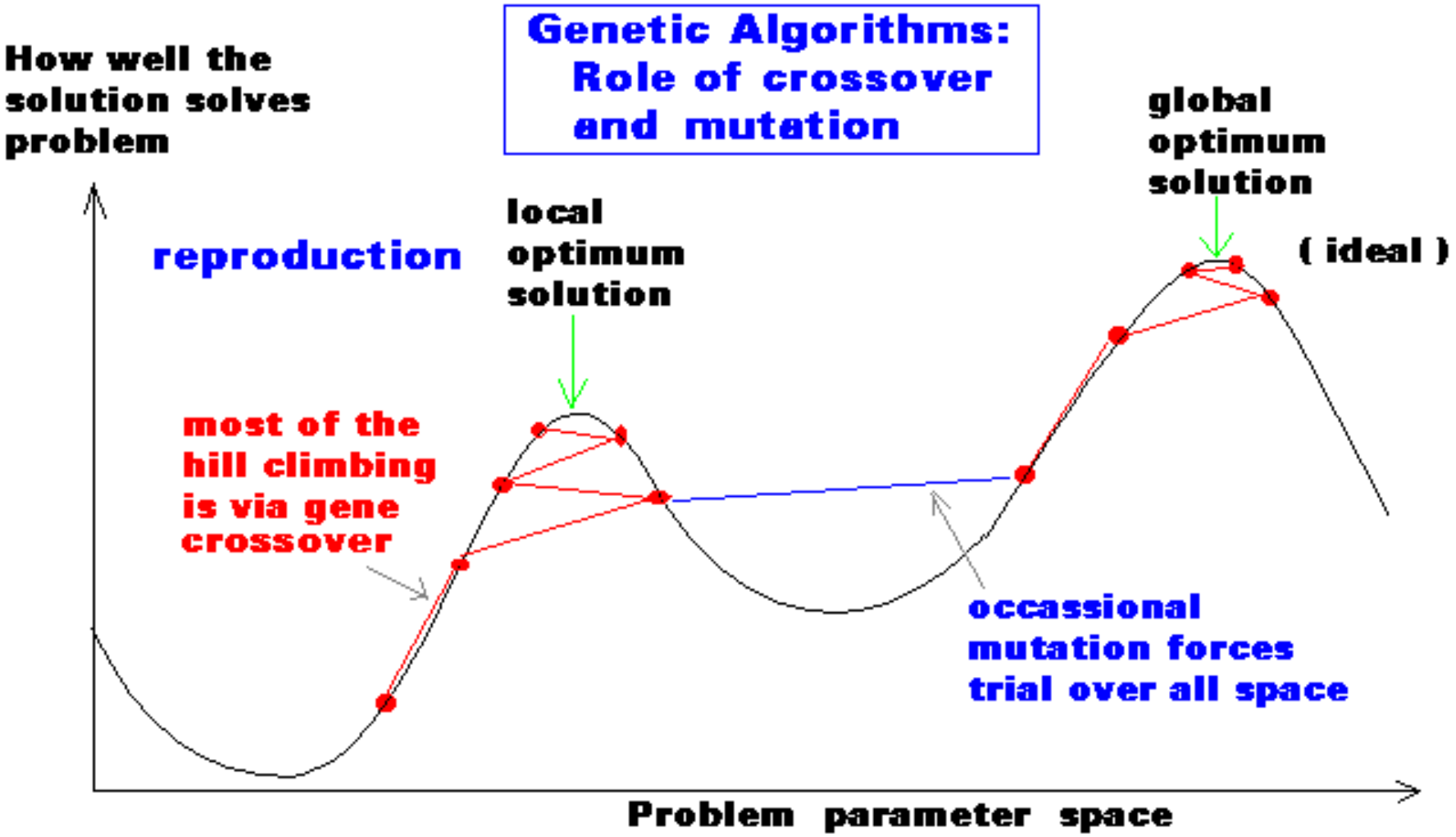
Genetic algorithm: mutation

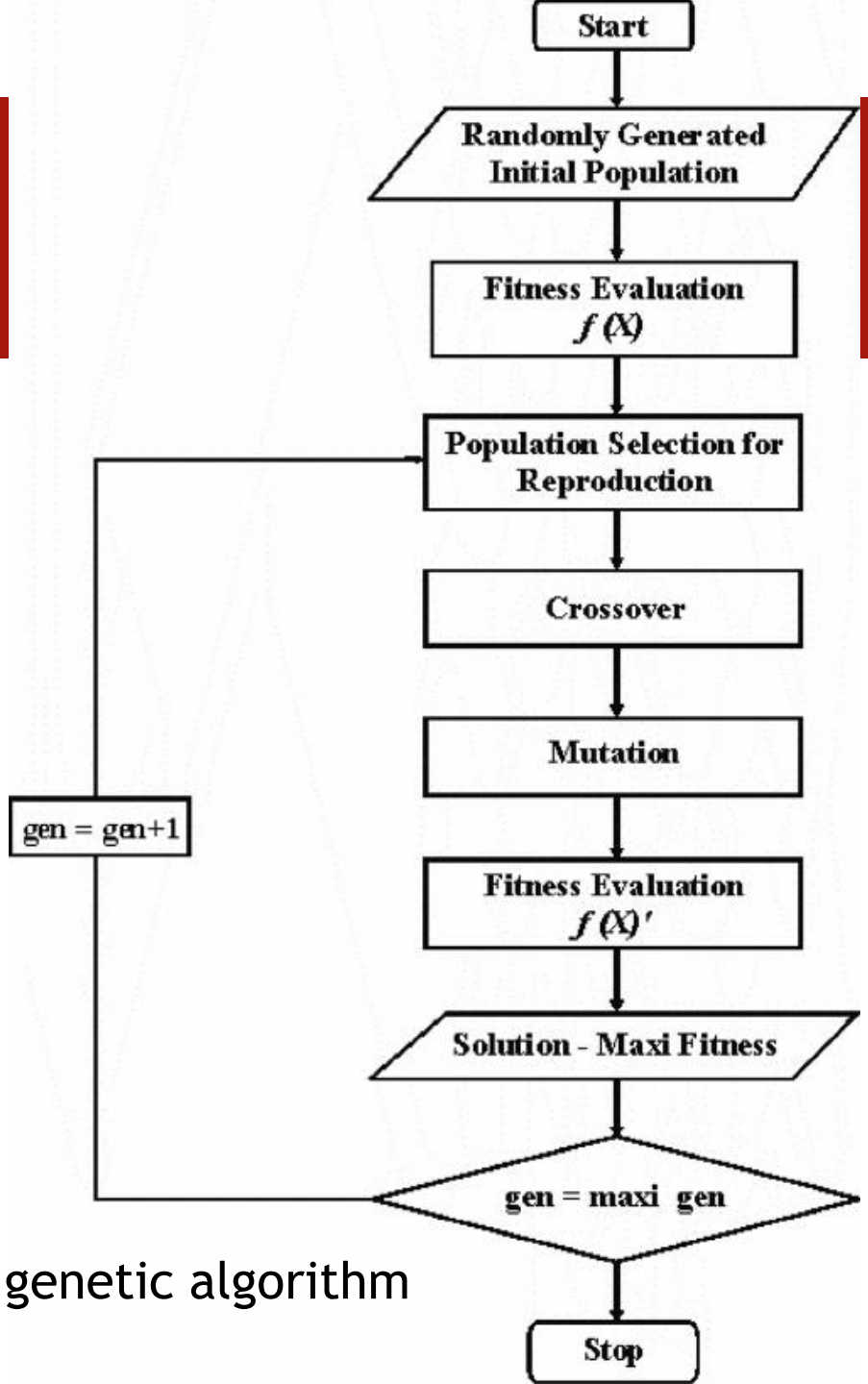
Mutation involves an occasional, random change in the value of an element of a code string. In the case of a binary code, this simply means changing a one to a zero and vice versa.

A1 = 0 1 1 0 1

A1’ = 0 1 0 0 1

Genetic algorithms





The scheme of genetic algorithm