

Genetic Algorithm

A decorative graphic consisting of a thin blue curved line starting from the left edge and a larger, solid blue curved shape in the bottom right corner, both pointing towards the text.

A genetic algorithm (or GA) is a search technique used in computing to find true or approximate solutions to optimization and search problems.

(GA)s are categorized as global search heuristics.

(GA)s are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (recombination)

The GA is a probabilistic intelligent search algorithm, which searches a population of points in parallel.

Basically, GA differs from other traditional optimization methods in three significant points:


- It searches a population of points in parallel
 - It uses probabilistic rules rather than deterministic ones
 - It can process an encoding set of parameters
-

The evolution usually starts from a population of randomly generated individuals and happens in generations.

In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified to form a new population.

The new population is used in the next iteration of the algorithm.

The algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.



**No convergence rule
or guarantee!**

Evolution process

The GA searches for the global optimum value of the objective function through a search space, which is called **population**.

The population is constituted from a number of possible solutions known as **individuals**.

Each individual, which is also called **chromosome**, represents a definite solution

Vocabulary

Individual

Any possible solution

Population

Group of all individuals Target
(search space)

Fitness

function that we optimize (each
individual has a fitness)



Basic Genetic Algorithm

- Start with a large “population” of randomly generated “attempted solutions” to a problem
 - Repeatedly do the following:
 - ✓ Evaluate each of the attempted solutions
 - ✓ (probabilistically) keep a subset of the best solutions
 - ✓ Use these solutions to generate a new population
 - Quit when you have a satisfactory solution (or you run out of time)
-

To initiate the population, a number of individuals are randomly formulated in the range of variables

Each individual in the randomly-created population is tested for violating the constraints

If the solution is infeasible, a suitable additional penalty cost is assigned to the individual

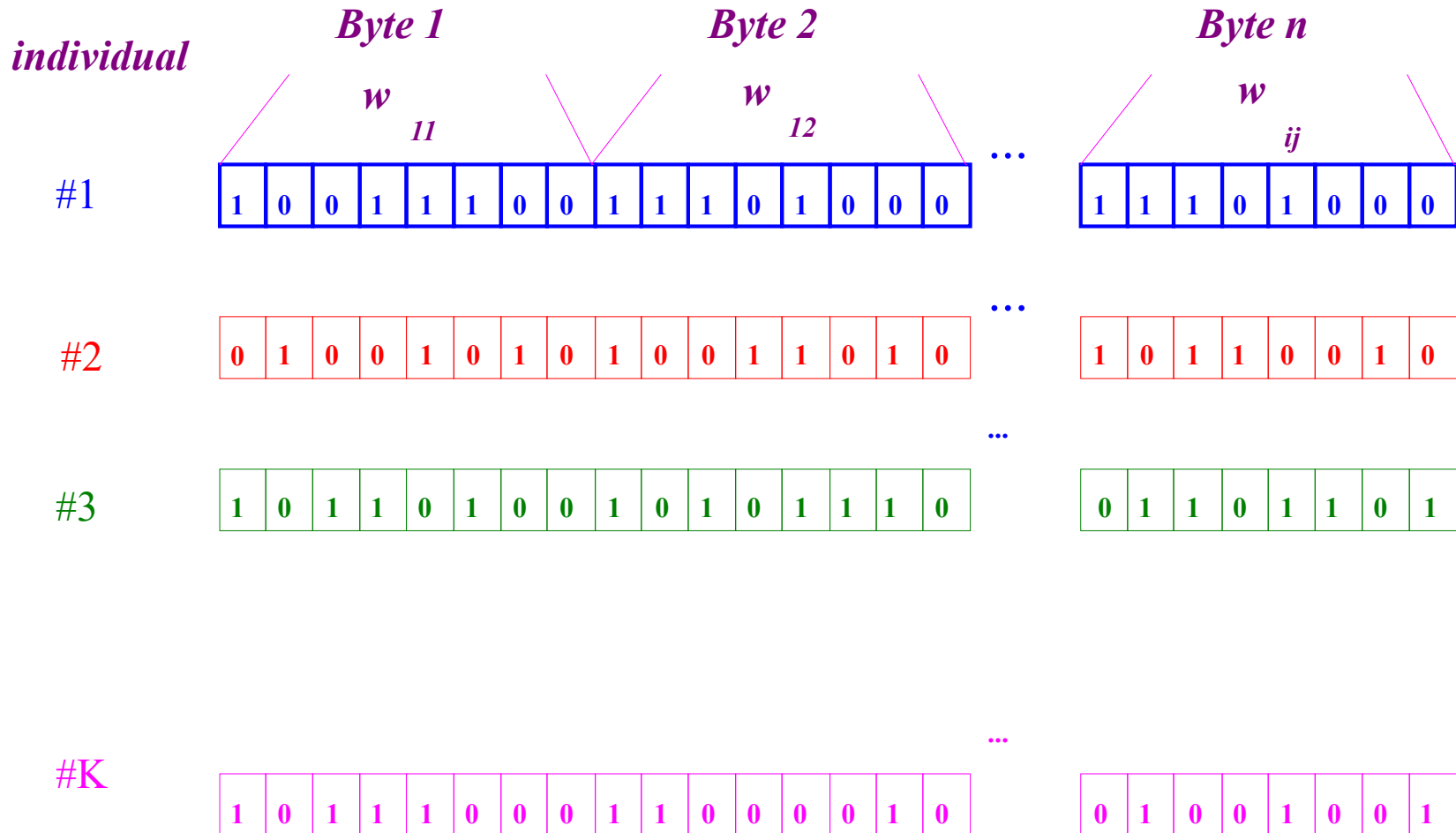
The performance of each individual is evaluated by calculating the value of objective function and adding the corresponding penalty term if exists

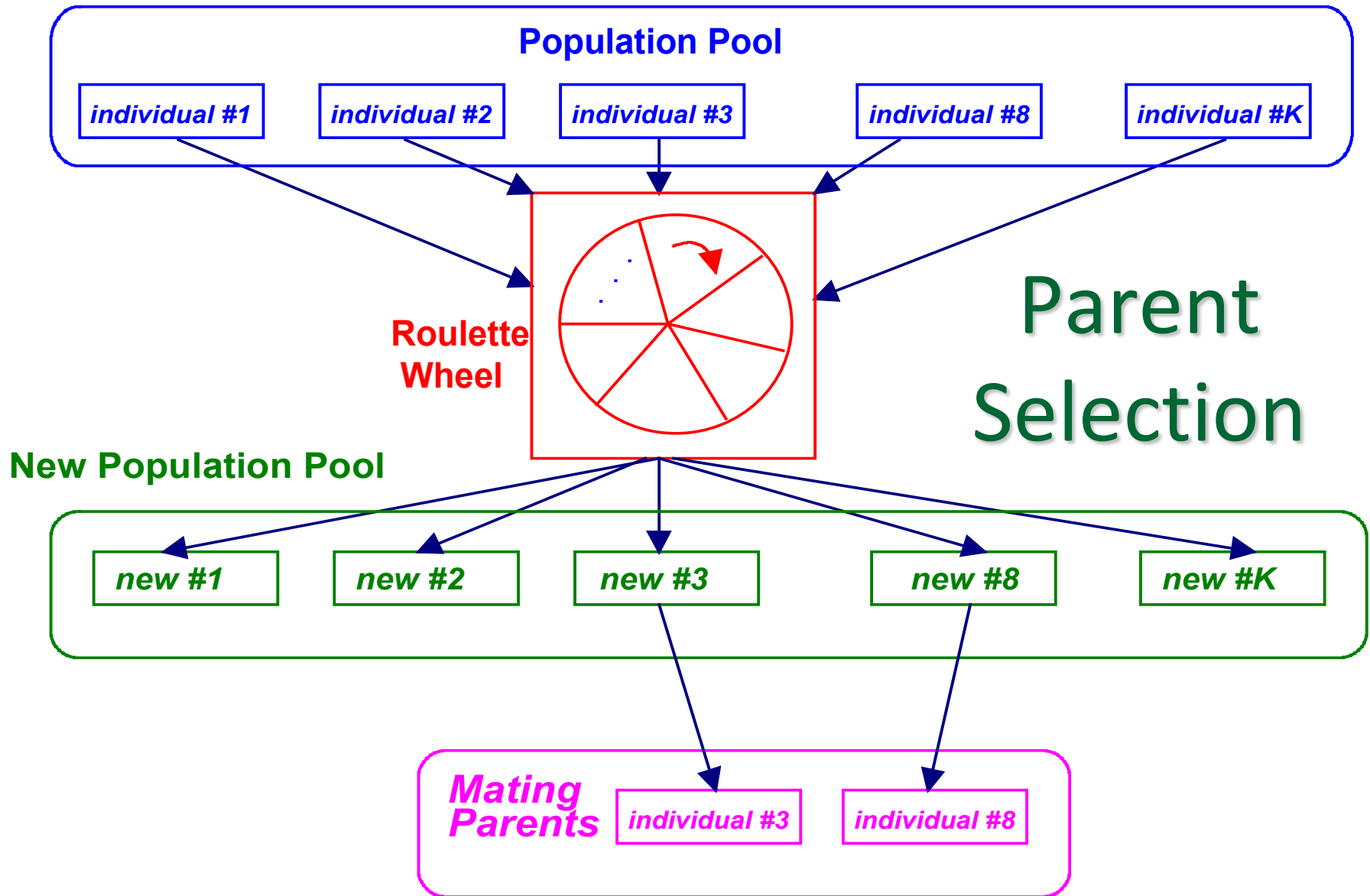
The individuals are then ranked depending on their corresponding costs and a suitable fitness value is assigned to each one

The fitness values could be calculated depending on the position of the individuals within the population rather than their distinct performance

Fitness values between maximum and minimum limits are calculated with fixed incremental steps and assigned to the ranked individuals

Population Pool





Constraints representation

To employ GA with a tightly-constrained problem, it is possible to generate only feasible solutions by avoiding individuals which violate the given constraints

Since the infeasible solutions mostly cover the search space at the initial generation, the complete avoidance of the infeasible solutions gives a high possibility for missing the area of global minimum

Another approach is to move the infeasible individuals to the nearest feasible area.

Constraints representation

For the highly-constrained problem, this would be too complex and a very time consuming process.

The penalty function approach is another alternative that converts the constrained problem to an unconstrained one by augmenting additional cost terms with the main objective function

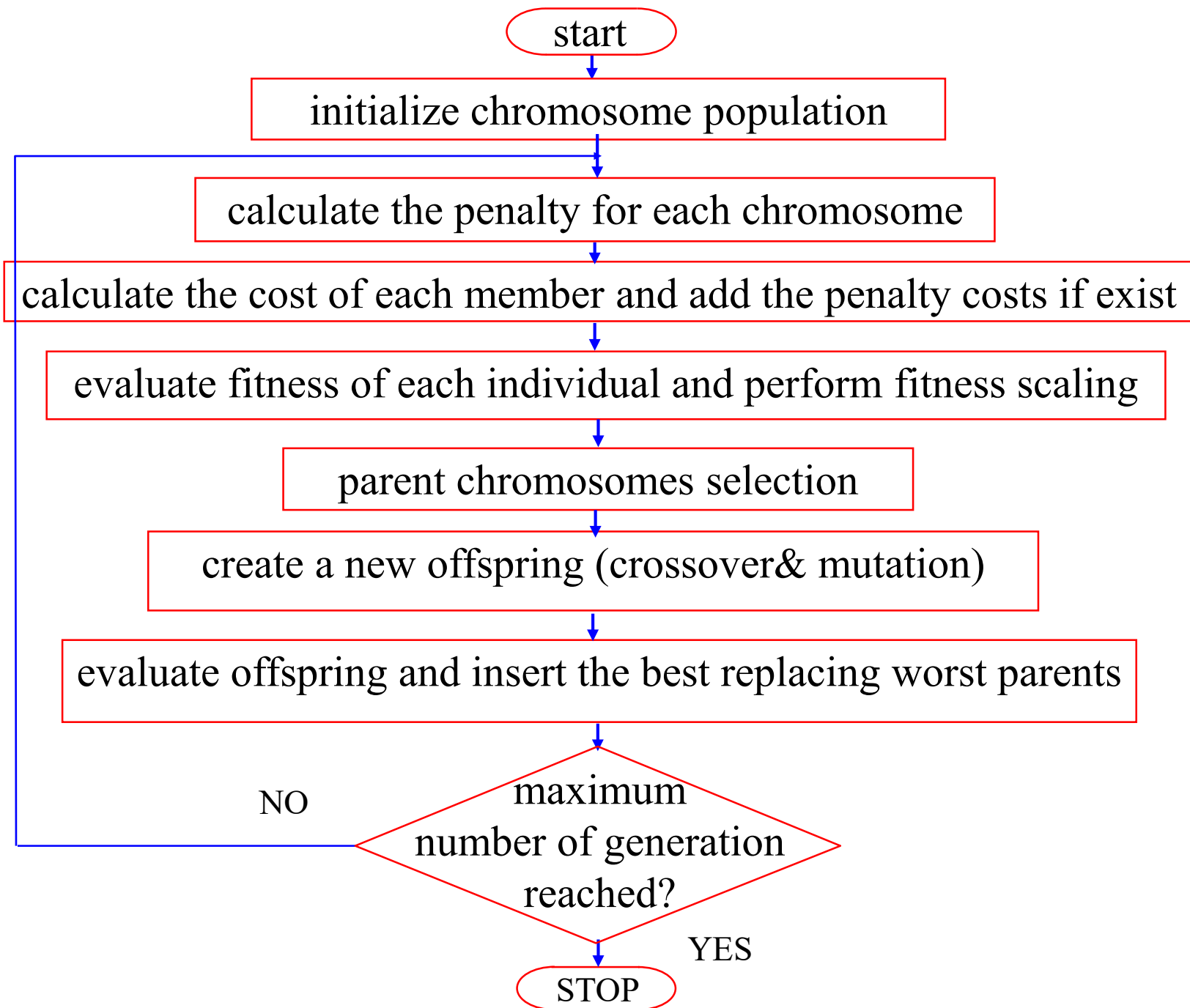
The additional terms assign nonlinear costs for solutions that violate the constraints depending on their relative locations with respect to the feasibility boundaries

Constraints representation

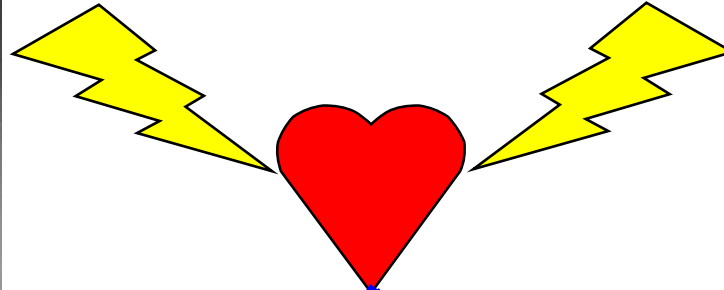
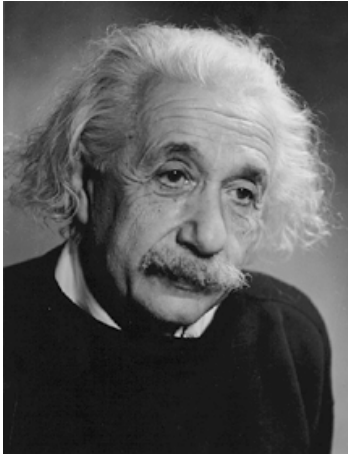
The adequate choice of the penalty functions and their parameters is an essential requirement to achieve rapid rejection of the infeasible solutions

It is necessary for this approach to differentiate in performance between the infeasible individuals themselves, which will help in the evolution process

To ensure fast rejection of the solutions that violate the constraints, a higher cost value has to be assigned to any infeasible solution than the feasible members



Cross Over



11111111	00000001
----------	----------

Smartness

Beauty

00000001	11111111
----------	----------

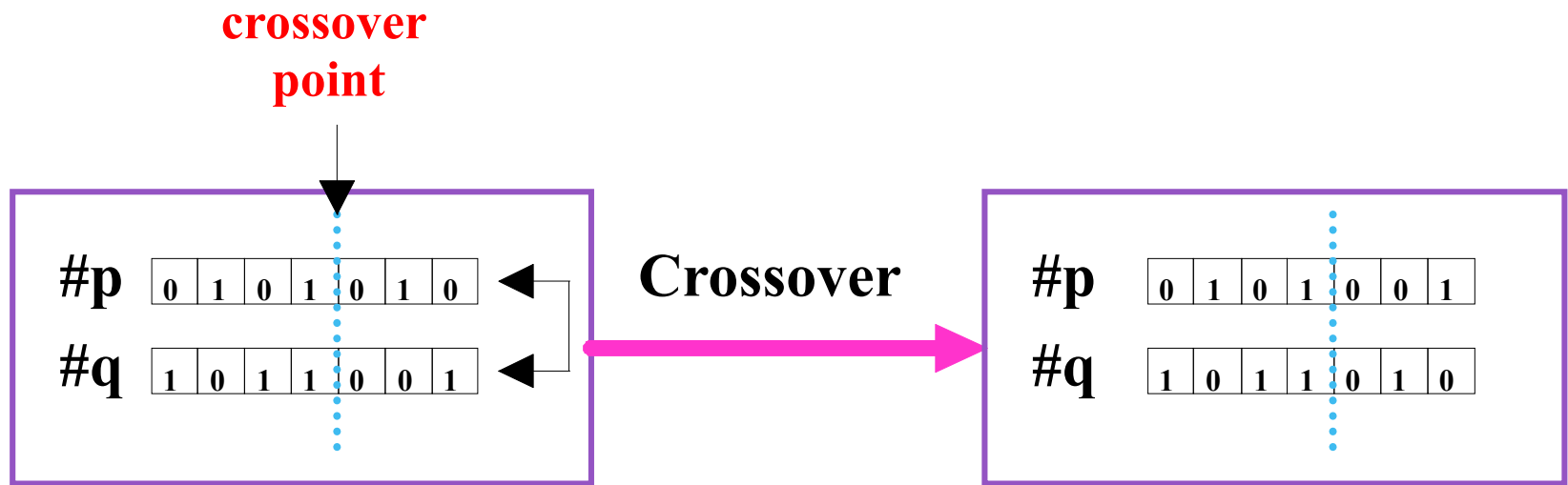
00000001	00000001
----------	----------

11111111	11111111
----------	----------



Global Optimality

Cross Over



Two-point crossover

Avoids cases where genes at the beginning and end of a chromosome are always split

Parents:

1010	0011	10	0011	0100	10
------	------	----	------	------	----

Offspring:

1010	0100	10	0011	0011	10
------	------	----	------	------	----



Uniform crossover

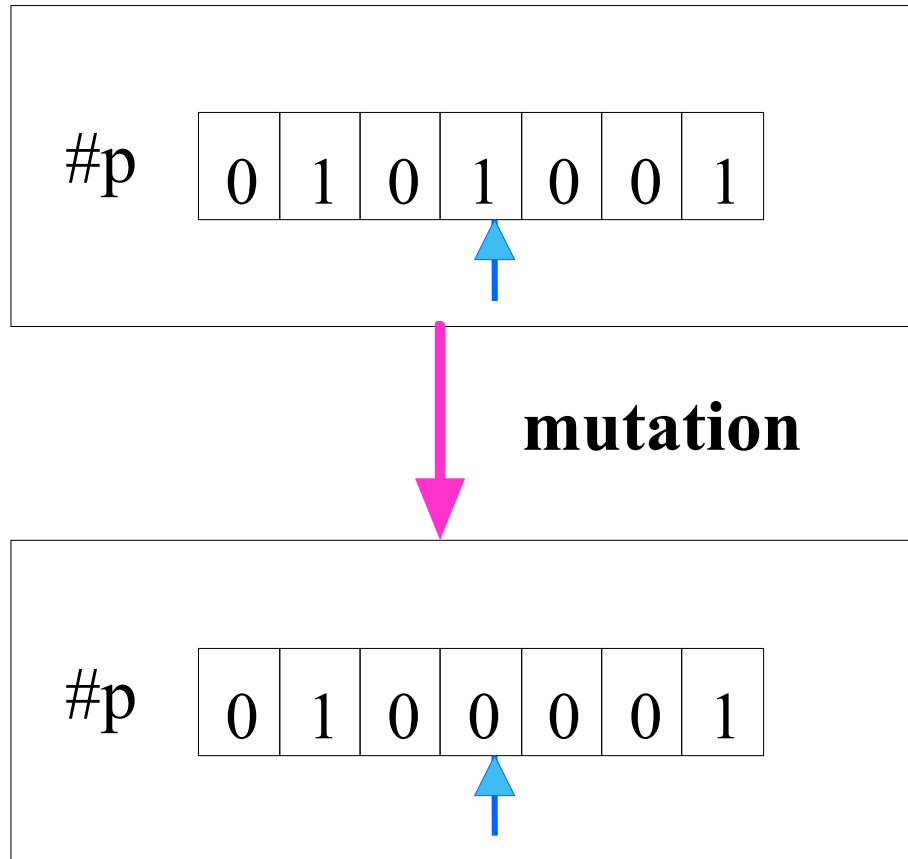
A random subset is chosen

The subset is taken from parent 1 and the other bits from parent 2.

Parents: 1010001110 0011010010

Offspring: 0011001010 1010010110

Mutation



Example

Suppose we want to maximize the number of ones in a string of (L) binary digits

An individual is encoded (naturally) as a string of l binary digits

The fitness (f) of a candidate solution is the number of ones in its genetic code

We start with a population of “ n ” random strings.

Suppose that $L = 10$ and $n = 6$

Example

We toss a fair coin 60 times and get the following initial population:

$$s1 = 1111010101 \quad f(s1) = 7$$

$$s2 = 0111000101 \quad f(s2) = 5$$

$$s3 = 1110110101 \quad f(s3) = 7$$

$$s4 = 0100010011 \quad f(s4) = 4$$

$$s5 = 1110111101 \quad f(s5) = 8$$

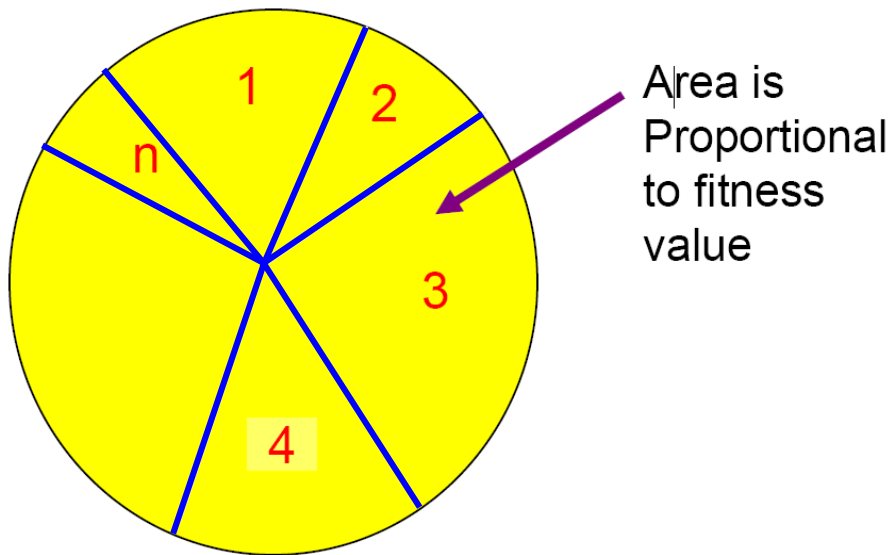
$$s6 = 0100110000 \quad f(s6) = 3$$

Example

We randomly select a subset of the individuals based on their fitness:

Individual “i” will have a probability to be chosen

$$\frac{f(i)}{\sum_i f(i)}$$



Example

Suppose that, after performing selection, we get the following population::

$$s1' = 1111010101 \quad (s1)$$

$$s2' = 1110110101 \quad (s3)$$

$$s3' = 1110111101 \quad (s5)$$

$$s4' = 0111000101 \quad (s2)$$

$$s5' = 0100010011 \quad (s4)$$

$$s6' = 1110111101 \quad (s5)$$

Example

- Next we mate strings for crossover. For each couple we first decide (using some pre-defined probability, for instance 0.6) whether to actually perform the crossover or not
 - If we decide to actually perform crossover, we randomly extract the crossover points, for instance 2 and 5
-

Example

Before crossover:

$$s_1' = \underbrace{11}_{\text{blue}} \underbrace{110}_{\text{red}} \underbrace{10101}_{\text{blue}} \quad s_2' = \underbrace{11}_{\text{red}} \underbrace{101}_{\text{blue}} \underbrace{110101}_{\text{red}}$$

After crossover:

$$s_1'' = \underbrace{1110110101}_{\text{blue}} \quad s_2'' = \underbrace{1111010101}_{\text{red}}$$

Mutations

The final step is to apply random mutations: for each bit that we are to copy to the new population we allow a small probability of error (for instance 0.1)

Initial strings

$s1'' = 1110110101$

$s2'' = 1111010101$

$s3'' = 1110111101$

$s4'' = 0111000101$

$s5'' = 0100011101$

$s6'' = 1110110011$

After mutating

$s1''' = 1110100101$

$s2''' = 1111110100$

$s3''' = 1110101111$

$s4''' = 0111000101$

$s5''' = 0100011101$

$s6''' = 1110110001$



Mutations

And now, iterate ...

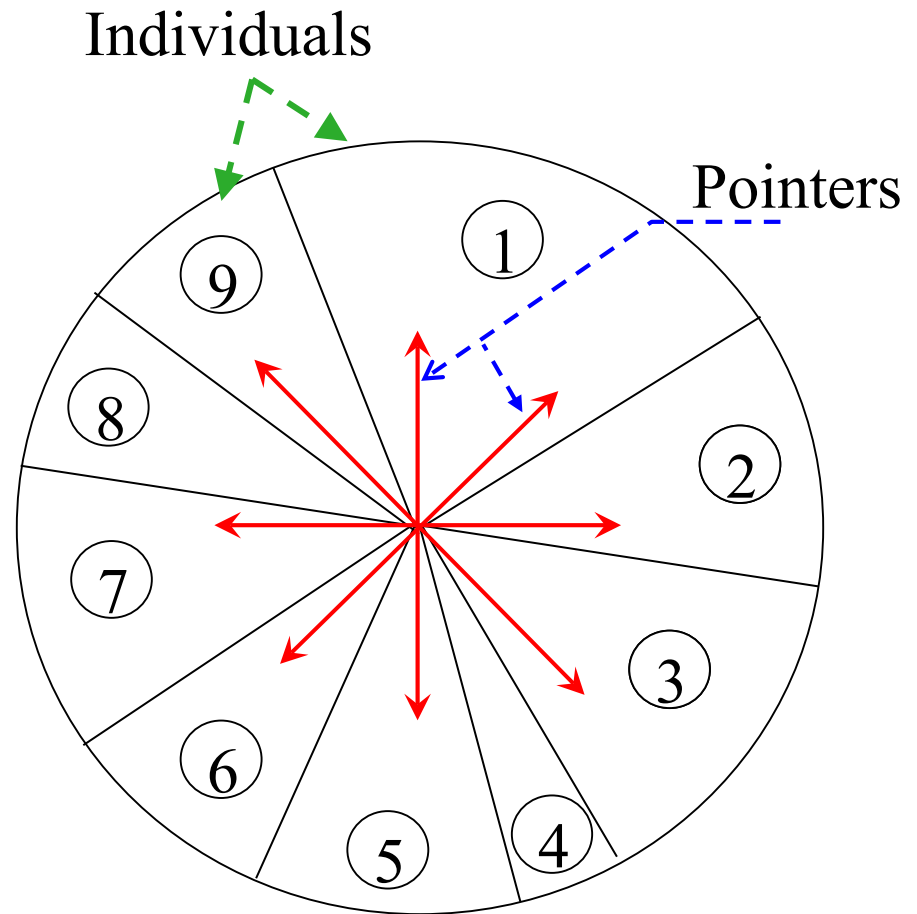
In one generation, the total population fitness changed from 34 to 37, thus improved by ~9%

At this point, we go through the same process all over again, until a stopping criterion is met



Selection

With “N” required selections, “N” equally spaced pointers can be used instead of a single selection pointer, which is employed in classical roulette wheel method



Recombination

New generations are produced by means of:

Crossover and mutation

Crossover is a genetic process to exchange information between members of the population.

For the real-valued encoding, the max-min arithmetical crossover operator can be used as follows:

$$G_1 = \alpha_s \cdot P_1 + (1 - \alpha_s) \cdot P_2$$

$$G_2 = (1 - \alpha_s) \cdot P_1 + \alpha_s \cdot P_2$$

Recombination

$$G_1 = \alpha_s \cdot P_1 + (1 - \alpha_s) \cdot P_2$$

$$G_2 = (1 - \alpha_s) \cdot P_1 + \alpha_s \cdot P_2$$

where: G_1, G_2 : the new generation “strings” of individuals

P_1, P_2 : the parents selected using the roulette wheel technique

α_s : a scaling factor

Mutation

It is the second process in the recombination, which is used to escape from possible local minima.

The mutation in the real-coded representations is accomplished by disturbing the gene values with low probability.

Elitism

Care has to be taken to ensure that the best solutions are not lost in moving from one generation to the next.

According to this strategy, which is known as 'Elitism' some of the fittest members of each generation are saved and copied into the next generation.

With this process, it is expected that the average fitness of the new generation is improved.



GA parameters

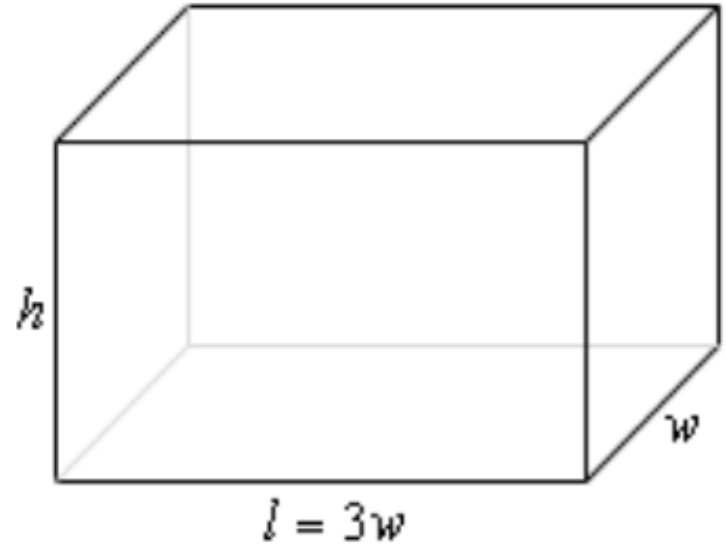
Number of individuals per subpopulation	20
Total population size	200
Generation gap	0.8
Insertion rate	0.9
Probability of crossover	0.9
Probability of mutation	0.01
Maximum generations	1000



EXAMPLE

We want to construct a box whose base length is 3 times the base width. The material used to build the top and bottom cost \$10/ft² and the material used to build the sides cost \$6/ft². If the box must have a volume of 50ft³ determine the dimensions that will minimize the cost to build the box

We want to minimize the cost of materials subject to the constraint that the volume must be 50 m^3 .



Note as well that the cost for each side is just the area of that side times the appropriate cost.

$$\text{Minimize : } C = 10(2lw) + 6(2wh + 2lh) = 60w^2 + 48wh$$

$$\text{Constraint : } 50 = lwh = 3w^2h$$

$$w = 1.8821$$

$$l = 3w = 3(1.8821) = 5.6463$$

$$h = \frac{50}{3w^2} = \frac{50}{3(1.8821)^2} = 4.7050$$

Also, even though it was not asked for, the minimum cost is :

$$C(1.8821) = \$637.60.$$
