Computational Intelligence

Samaneh Hosseini

Isfahan University of Technology

Outline

- What is Genetic Algorithm?
- Basic Structure
- Genotype Representation
- Population model
- Parent Selection
- Crossover
- Mutation
- ,.....

https://www.tutorialspoint.com/genetic_algorithms

Termination condition

- When there has been no improvement in the population for X iterations.
- When we reach an absolute number of generations.
- When the objective function value has reached a certain pre-defined value

Genetic algorithms: case study

- Let us find the maximum value of the function ($15x - x^2$)
- where parameter x varies between 0 and 15.
- For simplicity, we may assume that x takes only integer values.
- Thus, chromosomes can be built with only four genes:

Integer	*Binary code*	*Integer*	*Binary code*	*Integer*	*Binary code*
1	0 0 0 1	6	0 1 1 0	11	1 0 1 1
2	0 0 1 0	7	0 1 1 1	12	1 1 0 0
3	0 0 1 1	8	1 0 0 0	13	1 1 0 1
4	0 1 0 0	9	1 0 0 1	14	1 1 1 0
5	0 1 0 1	10	1 0 1 0	15	1 1 1 1

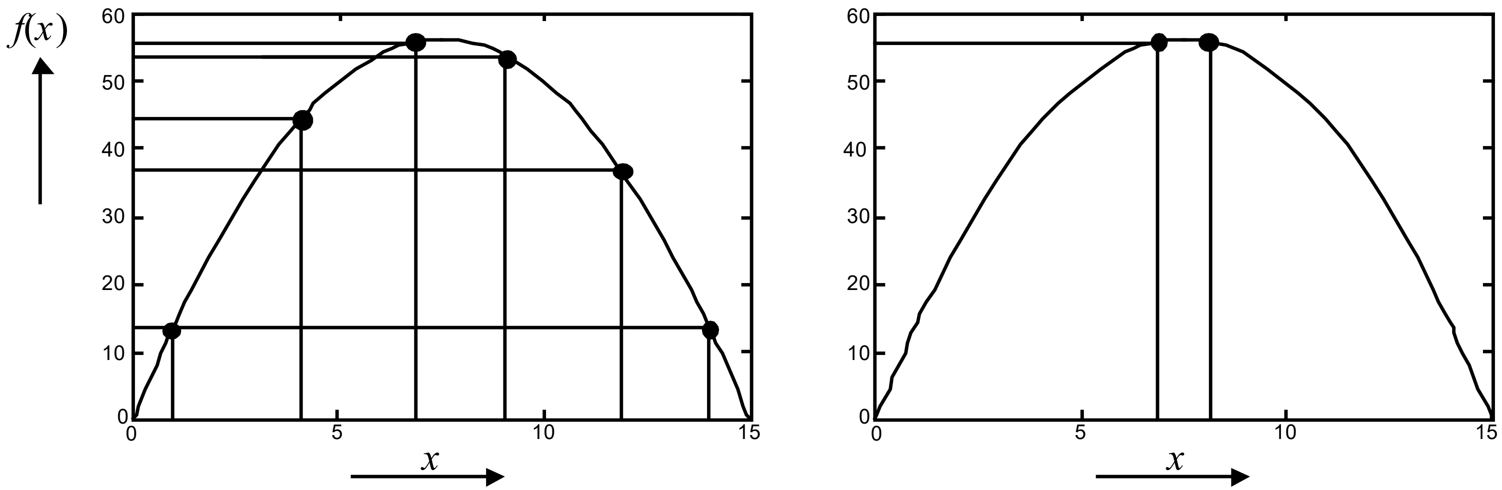
Genetic algorithms: case study

- Suppose that the size of the chromosome population N is 6,
- The crossover probability equals 0.7,
- The mutation probability equals 0.001.
- The fitness function in our example is defined by

$$f(x) = 15x - x^2$$

The fitness function and chromosome locations

Chromosome label	*Chromosome string*	*Decoded integer*	*Chromosome fitness*	*Fitness ratio, %*
X1	1 1 0 0	12	36	16.5
X2	0 1 0 0	4	44	20.2
X3	0 0 0 1	1	14	6.4
X4	1 1 1 0	14	14	6.4
X5	0 1 1 1	7	56	25.7
X6	1 0 0 1	9	54	24.8



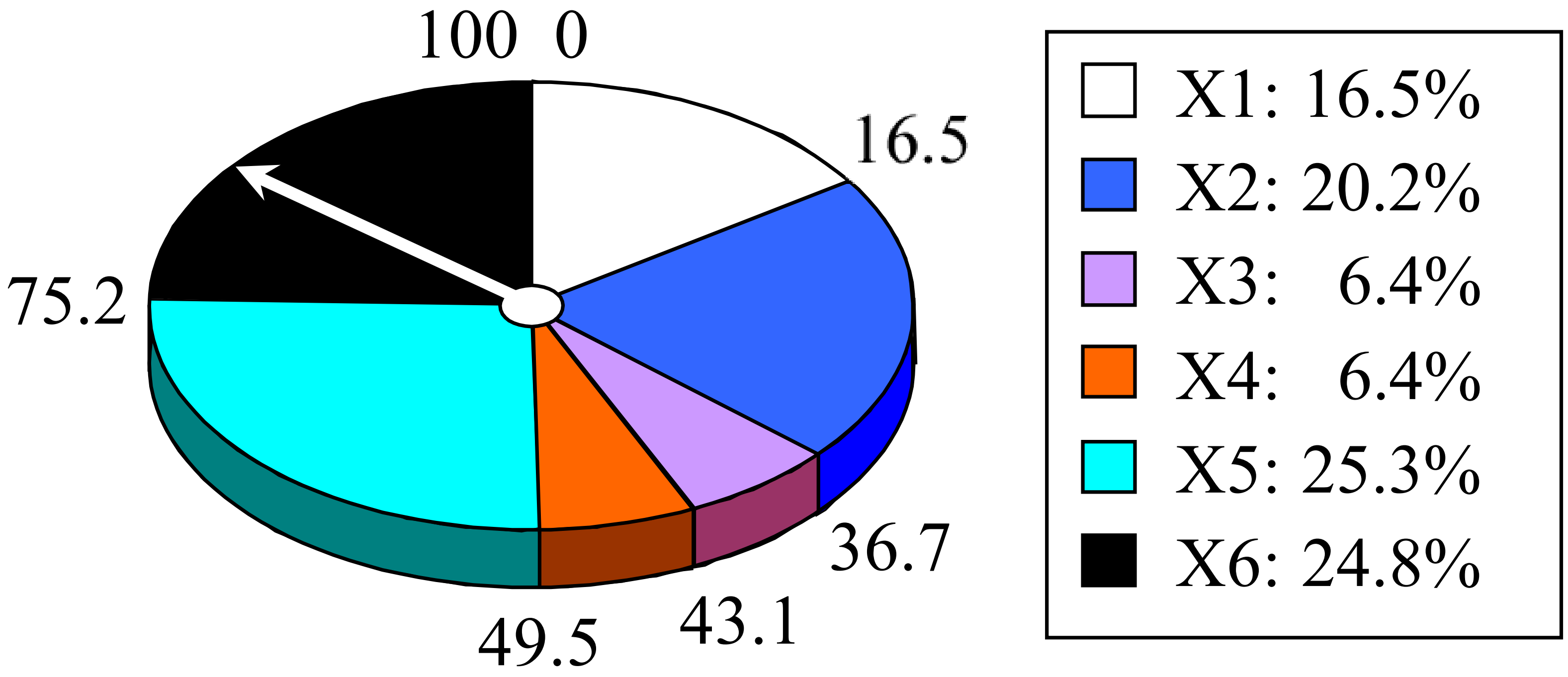
(*a*) Chromosome initial locations. (*b*) Chromosome final locations.

The fitness function and chromosome locations

- In natural selection, only the fittest species can survive, breed, and thereby pass their genes on to the next generation.
- GAs use a similar approach, but unlike nature, the size of the chromosome population remains unchanged from one generation to the next.
- The last column in Table shows the ratio of the individual chromosome’s fitness to the population’s total fitness.
- This ratio determines the chromosome’s chance of being selected for mating.
- The chromosome’s average fitness improves from one generation to the next.

Roulette wheel selection

- The most commonly used chromosome selection techniques is the roulette wheel selection.



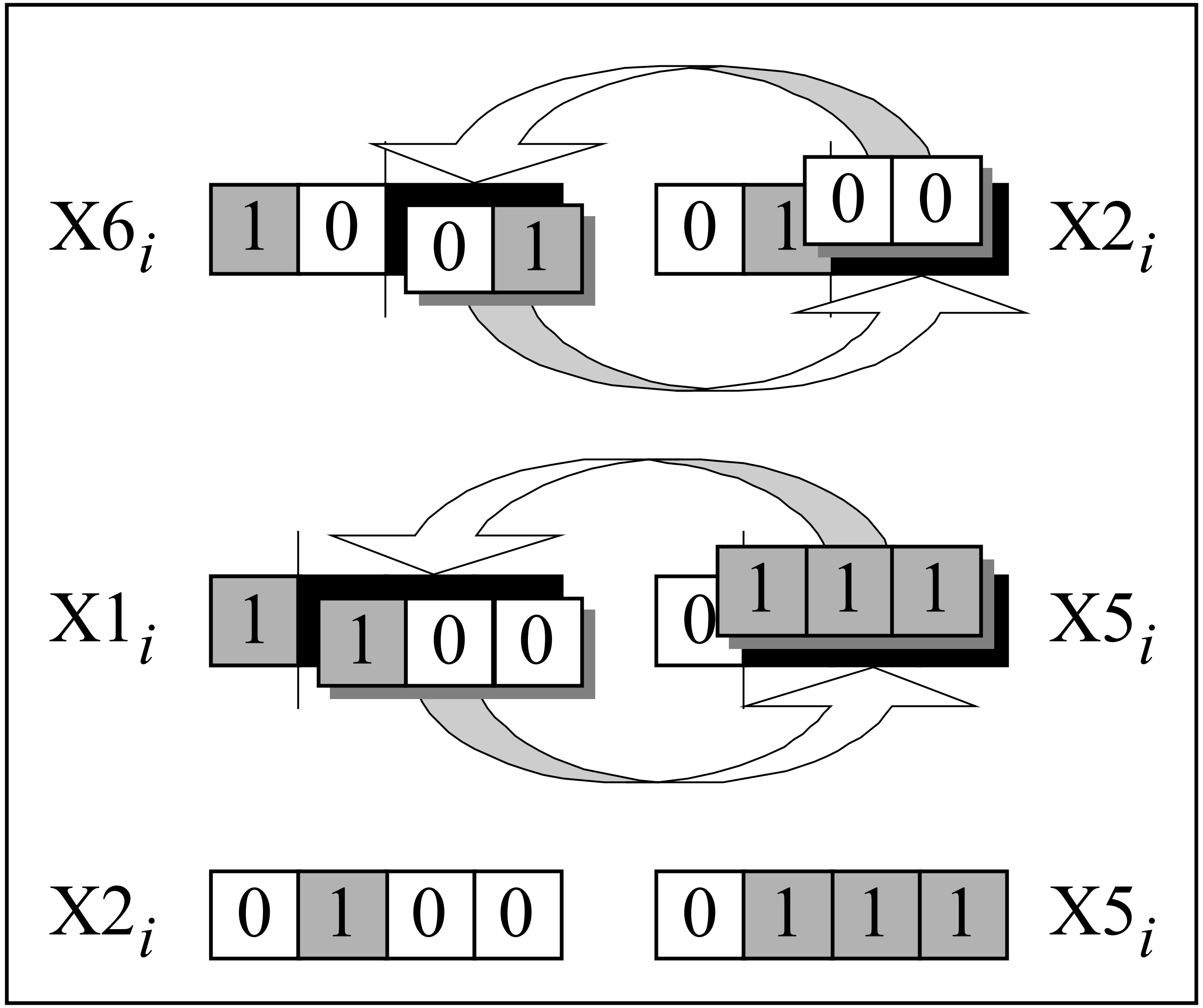
Crossover operator

- In our example, we have an initial population of 6 chromosomes.
- Thus, to establish the same population in the next generation, the roulette wheel would be spun six times.
- Once a pair of parent chromosomes is selected, the crossover operator is applied.

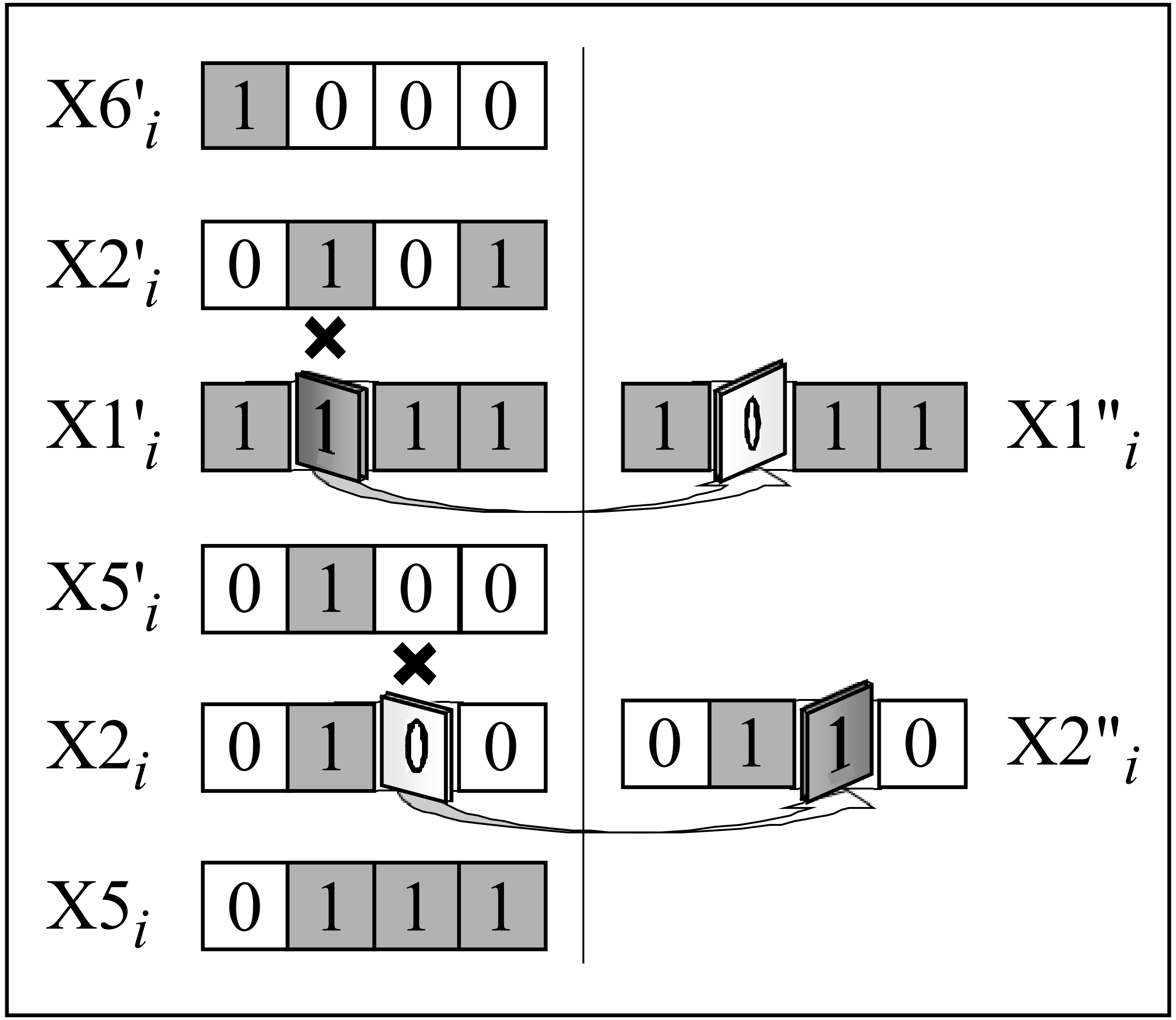
Crossover operator

- First, the crossover operator randomly chooses a crossover point where two parent chromosomes “break”, and then exchanges the chromosome parts after that point.
- As a result, two new offspring are created.
- If a pair of chromosomes does not cross over, then the chromosome cloning takes place, and the offspring are created as exact copies of each parent.

Crossover

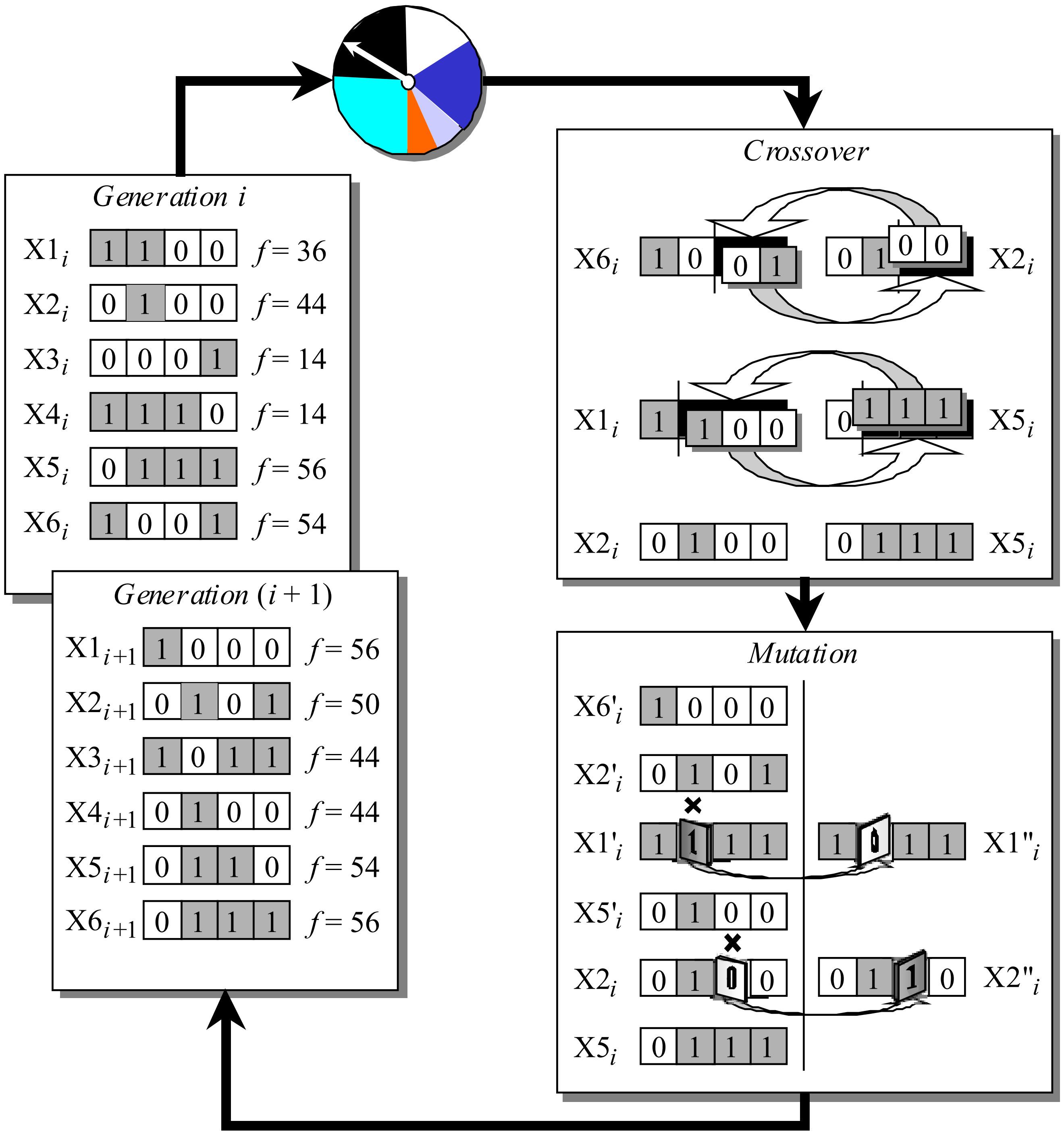


Mutation



X6'i
1 0 0 0
X2'i
0 1 0 1
X1'i
1 1 1 1
1 0 1 1
X1"i
X5'i
0 1 0 0
X2i
0 1 0 0
0 1 1 0
X2"i
X5i
0 1 1 1

The genetic algorithm cycle



Genetic algorithms: case study

- Suppose it is desired to find the maximum of the “peak” function of two variables:

$$f(x, y) = (1 - x)^2 e^{-x^2 - (y+1)^2} - (x - x^3 - y^3)\, e^{-x^2 - y^2}$$

- where parameters x and y vary between -3 and 3.

Genetic algorithms: case study

- The first step is to represent the problem variables as a chromosome -parameters x and y as a concatenated binary string:

$$\underbrace{1\ 0\ 0\ 0\ 1\ 0\ 1\ 0}_{x}\ \underbrace{0\ 0\ 1\ 1\ 1\ 0\ 1\ 1}_{y}$$

Genetic algorithms: case study

- We also choose the size of the chromosome population, for instance 6, and randomly generate an initial population.
- The next step is to calculate the fitness of each chromosome. This is done in two stages.
- First, a chromosome, that is a string of 16 bits, is partitioned into two 8-bit strings:

1	0	0	0	1	0	1	0	and	0	0	1	1	1	0	1	1

Genetic algorithms: case study

- Then these strings are converted from binary (base 2) to decimal (base 10):

$$(10001010)_2 = 1\times 2^7 + 0\times 2^6 + 0\times 2^5 + 0\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0 = (138)_{10}$$

and

$$(00111011)_2 = 0\times 2^7 + 0\times 2^6 + 1\times 2^5 + 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 = (59)_{10}$$

Genetic algorithms: case study

- Now the range of integers that can be handled by 8-bits, that is the range from 0 to (28 - 1), is mapped to the actual range of parameters x and y, that is the range from -3 to 3:

$$\frac{6}{256-1} = 0.0235294$$

- To obtain the actual values of x and y, we multiply their decimal values by 0.0235294 and subtract 3 from the results:

$$x = (138)_{10} \times 0.0235294 - 3 = 0.2470588$$

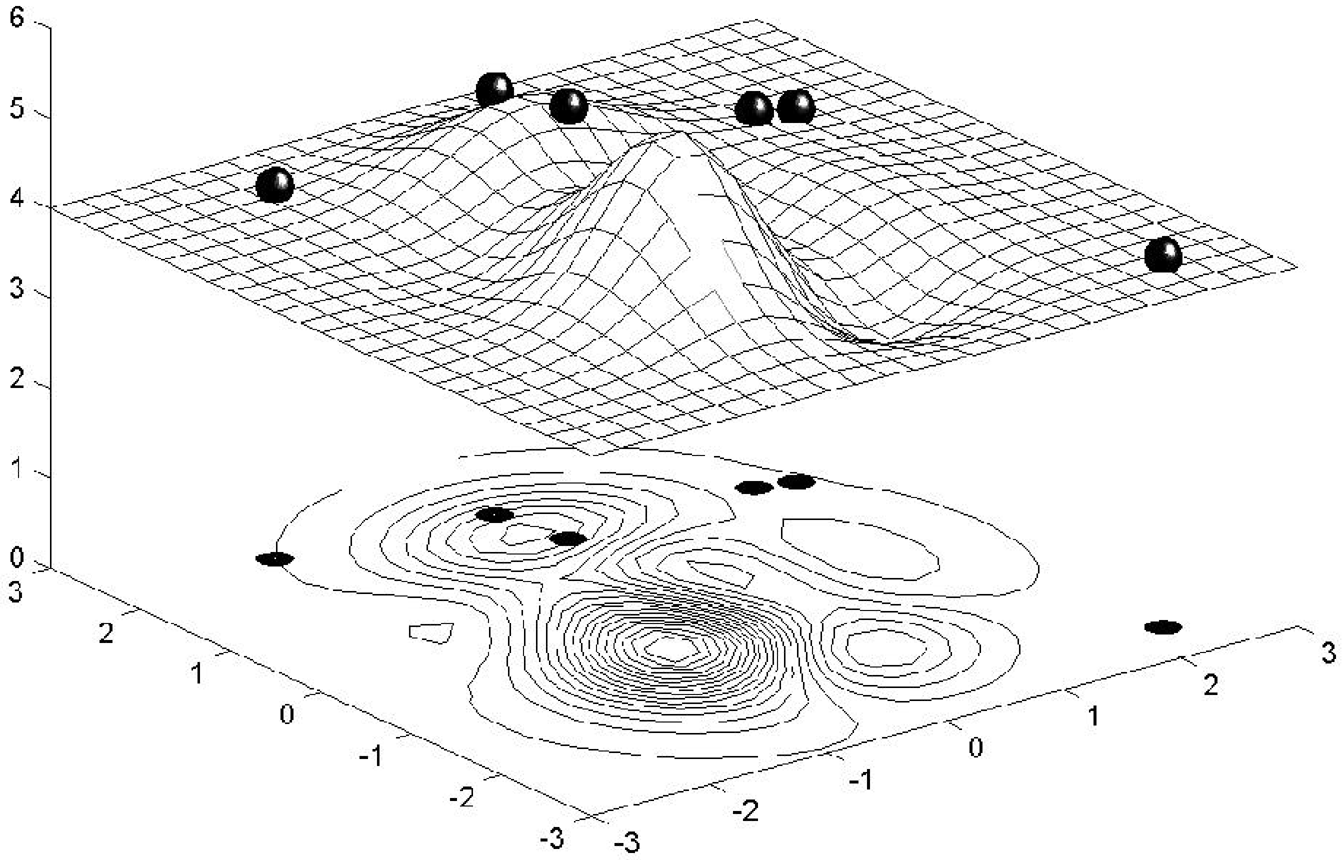
and

$$y = (59)_{10} \times 0.0235294 - 3 = -1.6117647$$

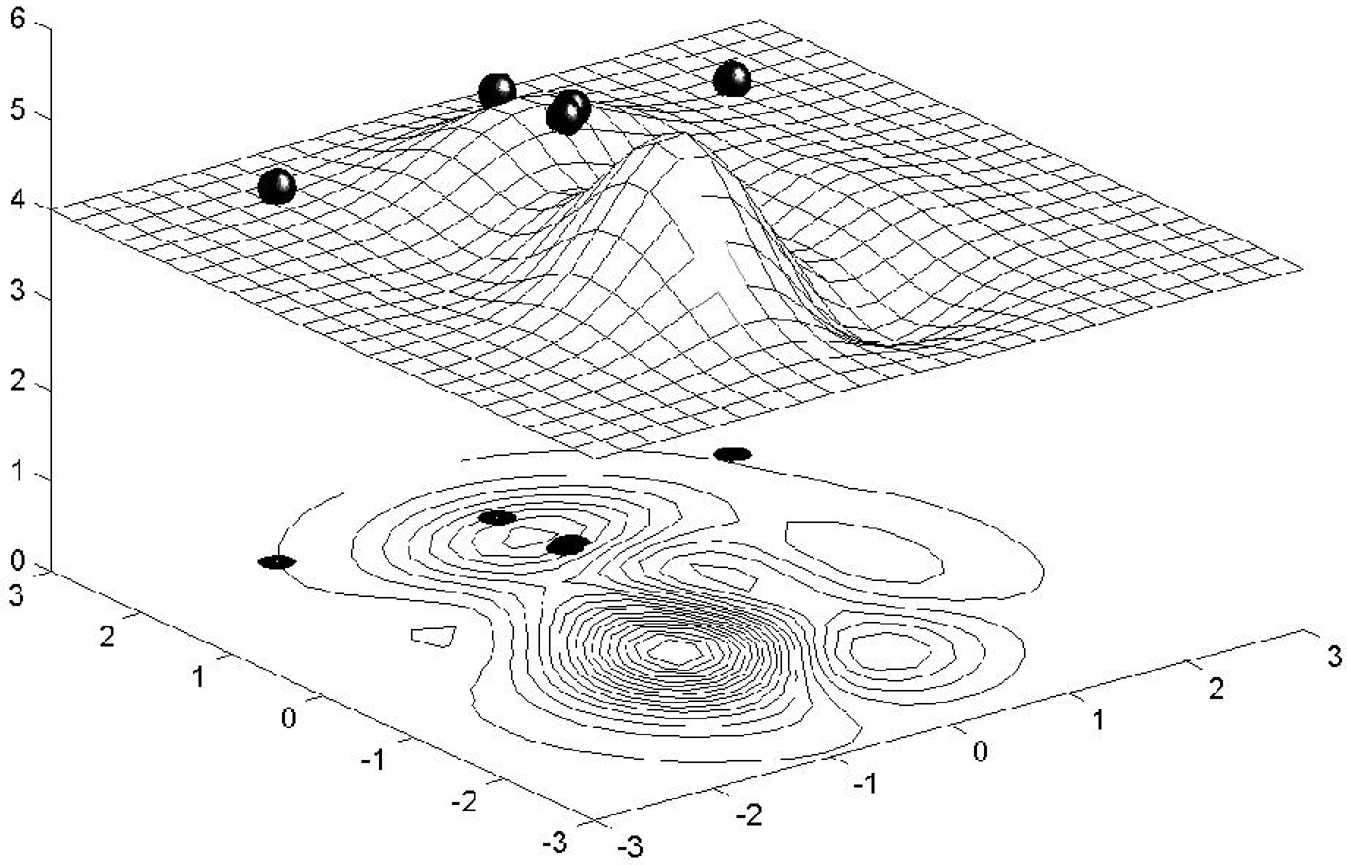
Genetic algorithms: case study

- Using decoded values of x and y as inputs in the mathematical function, the GA calculates the fitness of each chromosome.
- To find the maximum of the “peak” function, we will use crossover with the probability equal to 0.7 and mutation with the probability equal to 0.001.
- Suppose the desired number of generations is 100.
- That is, the GA will create 100 generations of 6 chromosomes before stopping.

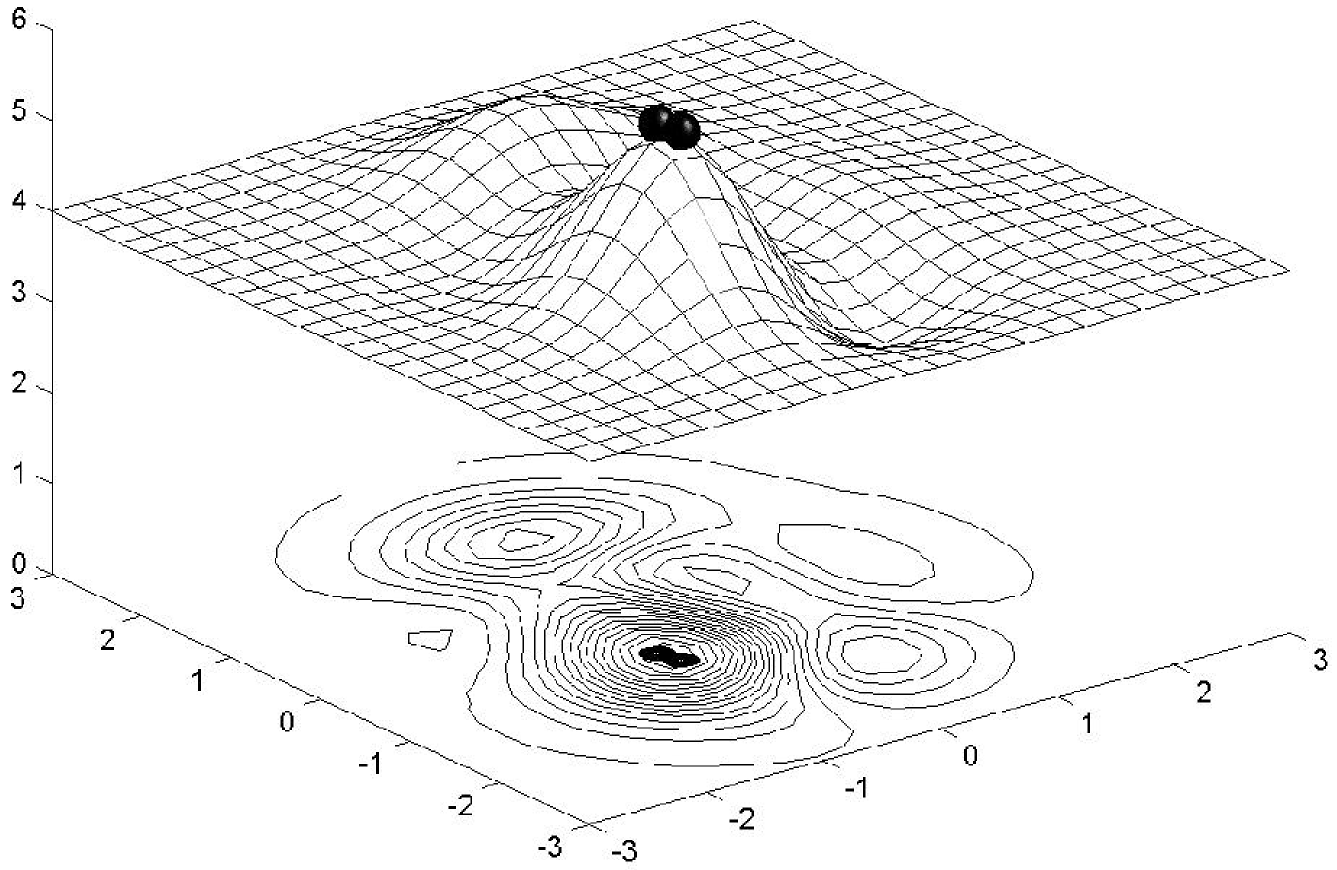
Chromosome locations on the surface of the "peak" function: initial population



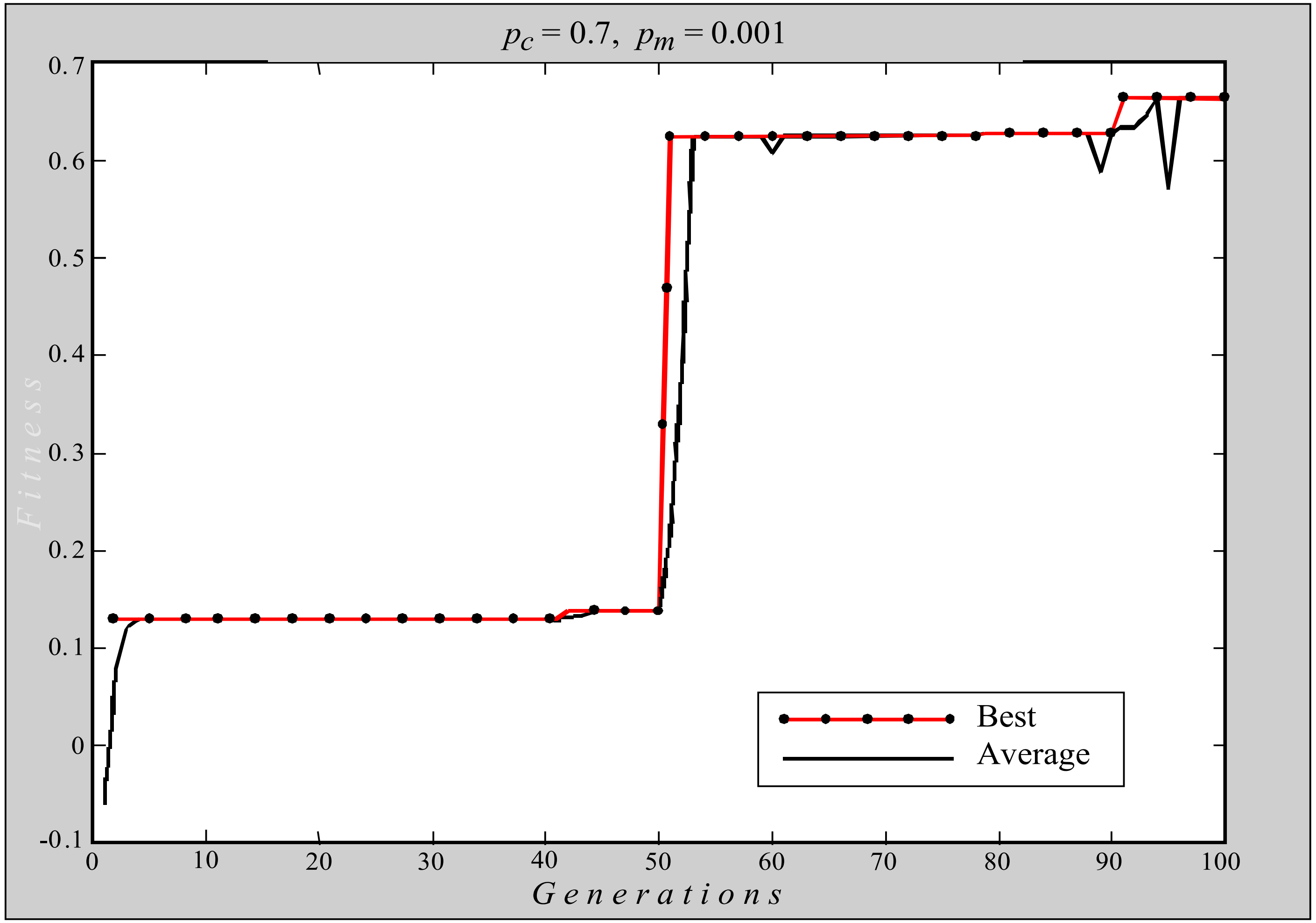
Chromosome locations on the surface of the “peak” function: initial population



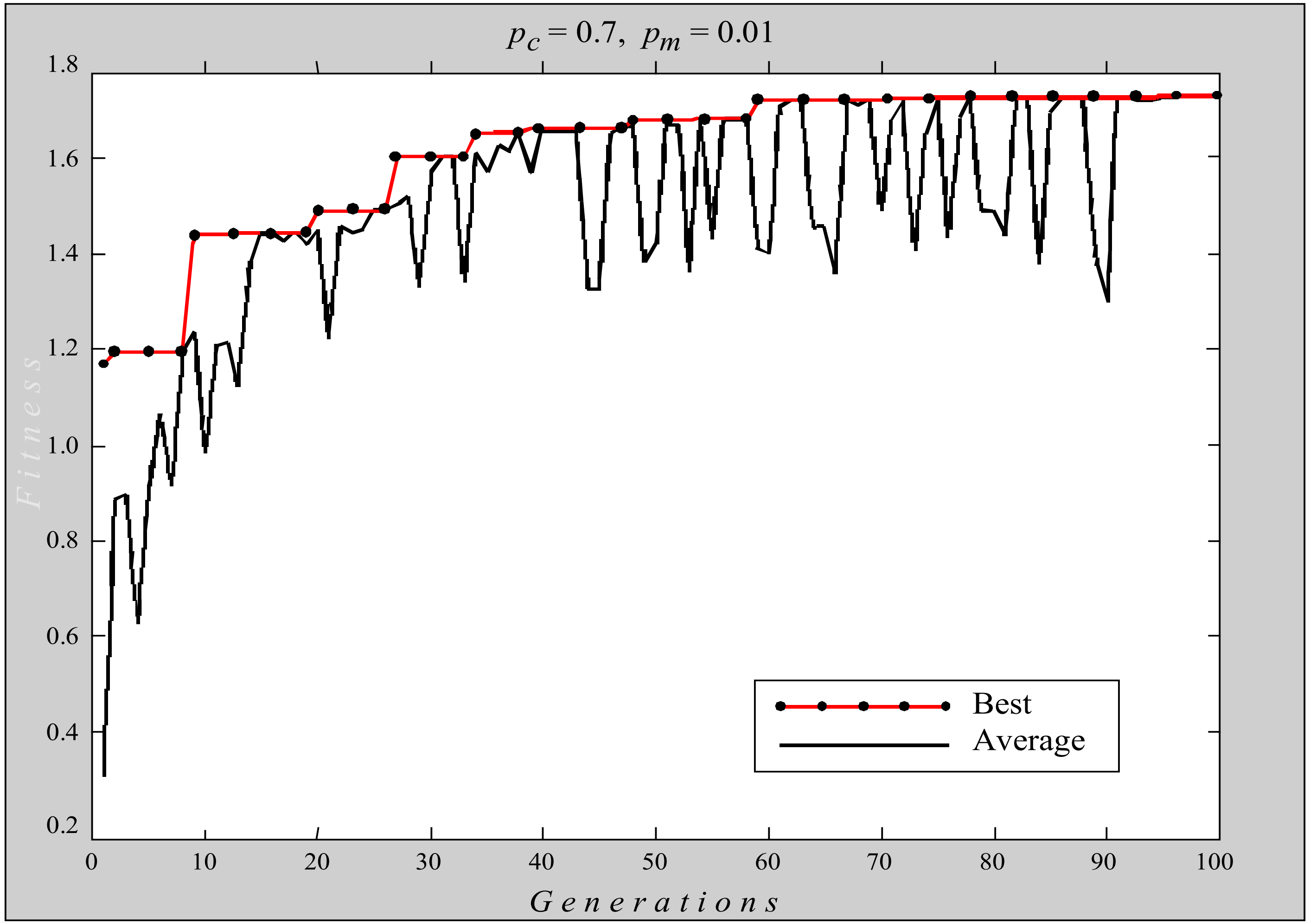
Chromosome locations on the surface of the "peak" function: initial population



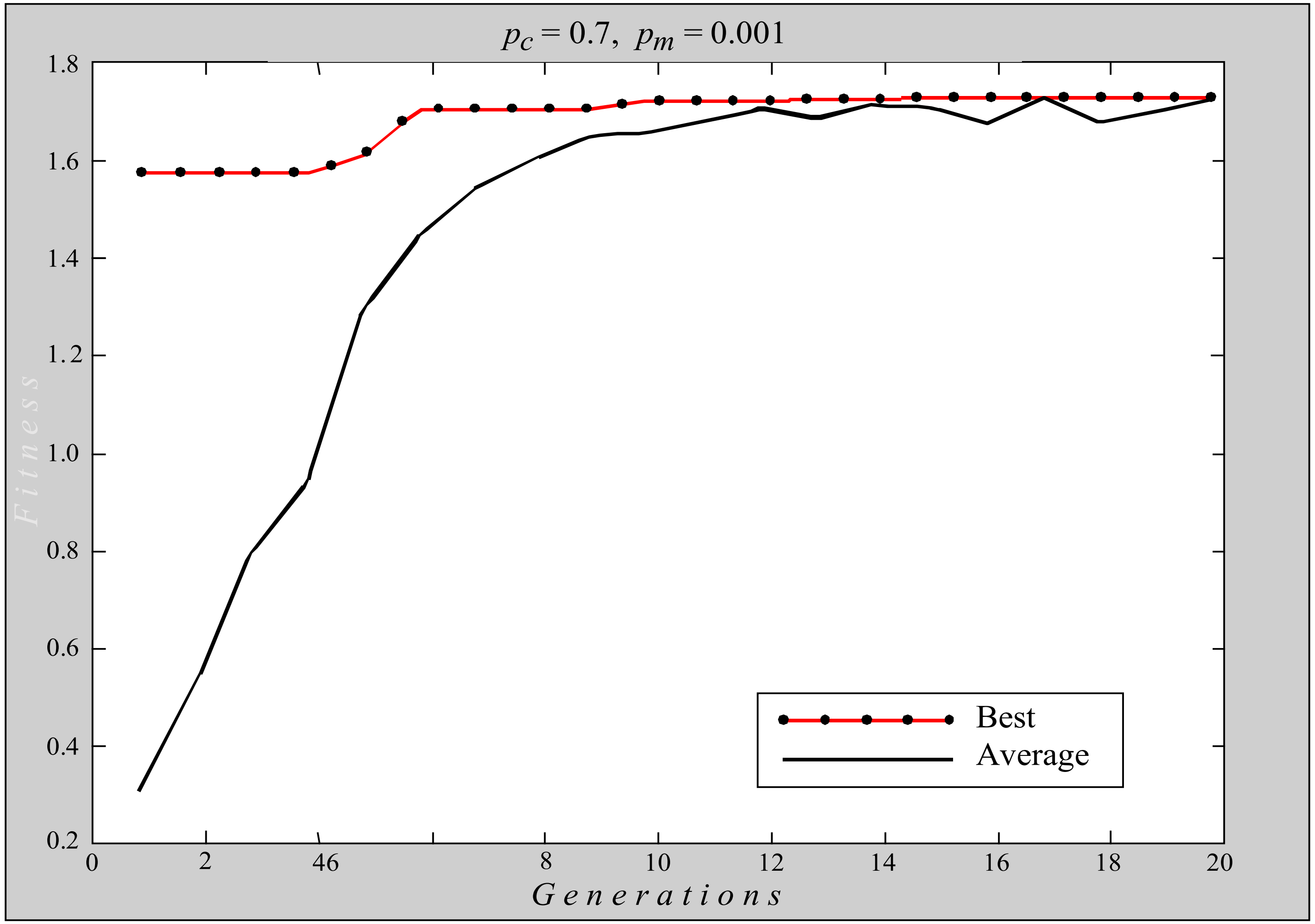
Performance graphs for 100 generations of 6 chromosomes: Local maximum



Performance graphs for 100 generations of 6 chromosomes: Global maximum



Performance graphs for 20 generations of 60 chromosomes

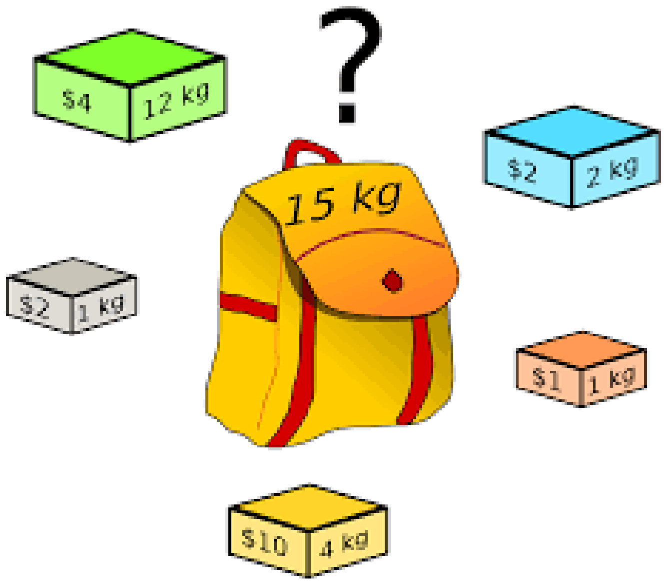


Steps in the GA development

1. Specify the problem, define constraints and optimum criteria;
2. Represent the problem domain as a chromosome;
3. Define a fitness function to evaluate the chromosome performance;
4. Construct the genetic operators;
5. Run the GA and tune its parameters.

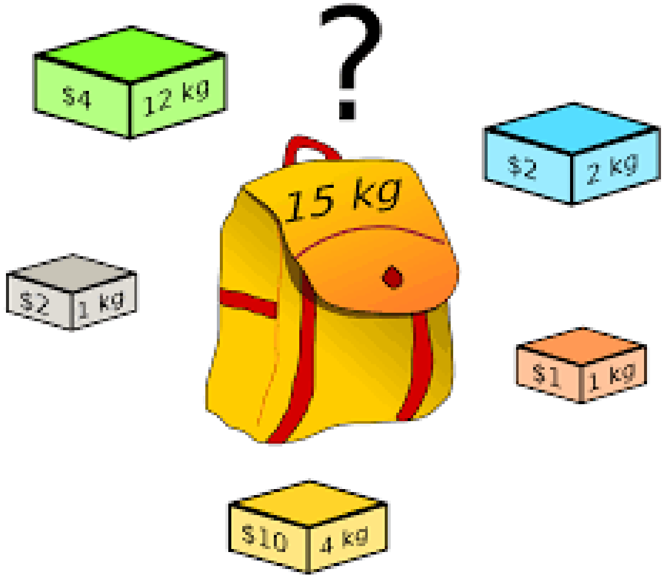
Case study: Knapsack Problem

- Problem Description:
  - You are going on a picnic.
  - And have a number of items that you could take along.
  - Each item has a weight and a benefit or value.
  - You can take one of each item at most.
  - There is a capacity limit on the weight you can carry.
  - You should carry items with max. values



Case study: Knapsack Problem

- Example:
  - Item: 1 2 3 4 5 6 7
  - Benefit: 5 8 3 2 7 9 4
  - Weight: 7 8 4 10 4 6 4
  - Knapsack holds a maximum of 22 pounds
  - Fill it to get the maximum benefit



Case study: Knapsack Problem

1. **[Start]**
   - ✓ Encoding: represent the individual.
   - ✓ Generate random population of *n* chromosomes (suitable solutions for the problem).
2. **[Fitness]** Evaluate the fitness of each chromosome.
3. **[New population]** repeating following steps until the new population is complete.
4. **[Selection]** Select the best two parents.
5. **[Crossover]** cross over the parents to form a new offspring (children).

Case study: Knapsack Problem

6. **[Mutation]** With a mutation probability.
7. **[Accepting]** Place new offspring in a new population.
8. **[Replace]** Use new generated population for a further run of algorithm.
9. **[Test]** If the end condition is satisfied, then **stop**.
10. **[Loop]** Go to step **2** .

Case study: Knapsack Problem: Start

- Encoding: 0 = not exist, 1 = exist in the Knapsack

    Chromosome: 1010110

Item.	1	2	3	4	5	6	7
Chro	1	0	1	0	1	1	0
Exist?	y	n	y	n	y	y	n

=> Items taken: 1, 3 , 5, 6.

- Generate random population of *n* chromosomes:
    a) 0101010
    b) 1100100
    c) 0100011

Case study:
Knapsack Problem: **Fitness & Selection**

a) 0101010: Benefit= 19, Weight= 24

Item	1	2	3	4	5	6	7
Chro	0	1	0	1	0	1	0
Benefit	5	8	3	2	7	9	4
Weight	7	8	4	10	4	6	4

b) 1100100: Benefit= 20, Weight= 19.

c) 0100011: Benefit= 21, Weight= 18.

=> We select Chromosomes b & c.

Case study:
Knapsack Problem: **Crossover & Mutation**

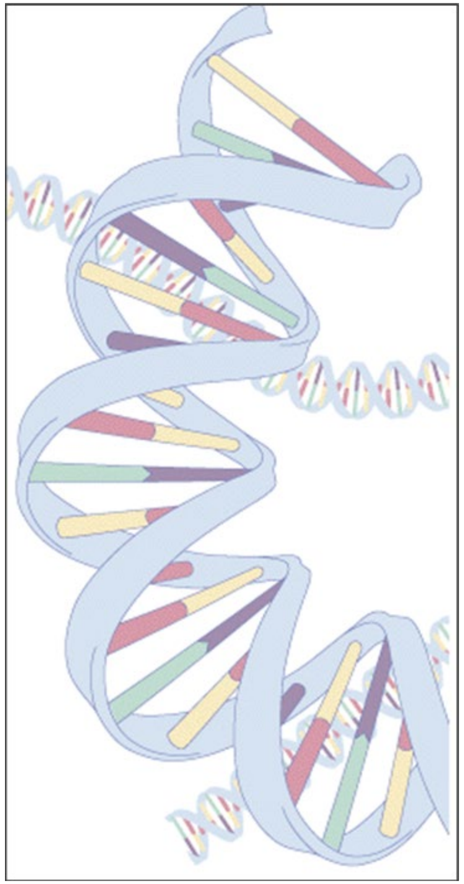
Parent 1 1 1 0 0 1 0 0

Parent 2 0 1 0 0 0 1 1

Child 1 1 1 0 0 0 1 1

Child 2 0 1 0 0 1 0 0

Mutation 0 1 0 0 1 0 1



Case study: Knapsack Problem: **Accepting, Replacing & Testing**

- Place new offspring in a new population.
- Use new generated population for a further run of algorithm.
- If the end condition is satisfied, then stop.
  End conditions:
  - Number of populations.
  - Improvement of the best solution.
- Else, return to step 2 [Fitness].