Computational Intelligence

Samaneh Hosseini

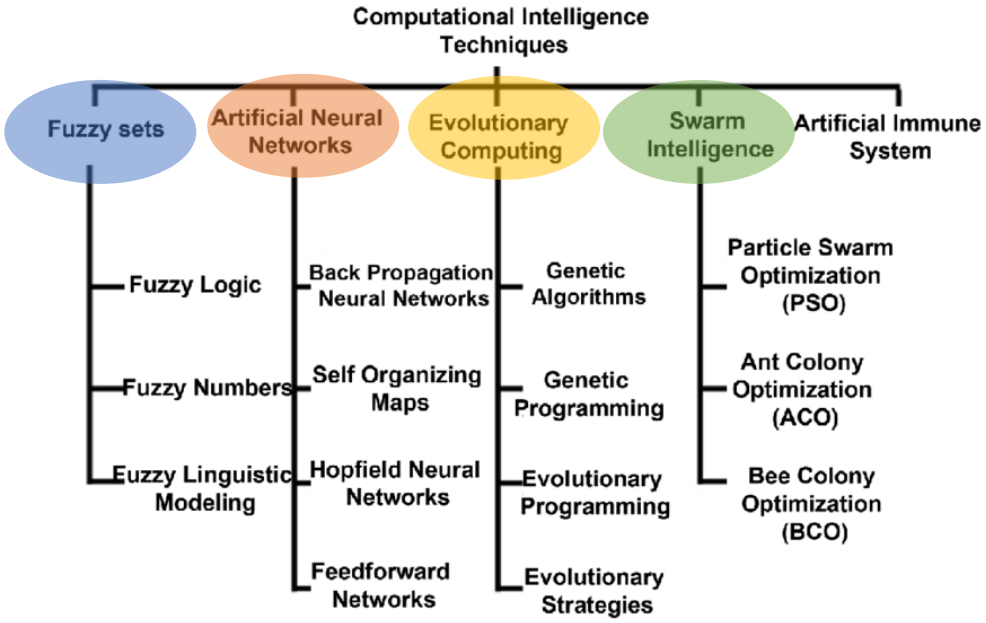
Isfahan University of Technology

Outline

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

https://www.tutorialspoint.com/genetic_algorithms

What is Computational Intelligence?



L1-Introduction
L2-Neuron math model
L3-Perceptron
L4-Building and Applying NN
L5-Gradient descent
L6-Vectorization
L7-Overfitting
L8-Regularization I
L9-Regularization II
L10-Optimization Algorithms I (mini-batches)
L11-Optimization Algorithms II(exponentially weighted averages)
L12-Hyperparameter Tuning
L13-Batch Normalization
L14-Soiftmax
L15-Convolutional Neural Networks
L16-Padding, Strided convolution
L17-Simple Convolutional Network
L18- Genetic Algorithms
L19- Genetic Algorithms
L20- Swarm Intelligence Introduction
L21-Particle Swarm Intelligence (PSO)
L22- PSO
L23-Discrete PSO
L24-PSO Hyper-parameters tuning
L25- Ant Colony Optimization (ACO)
L26- ACO
L27- Introduction
L28- Fuzzy Inference Systems
L-29 Fuzzy Inference Systems
L-30 Defuzzification

Genetic Algorithm

- Genetic Algorithm (GA) is a search-based optimization technique
- Based on the principles of Genetics and Natural Selection.
- It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve.

Optimization Problem

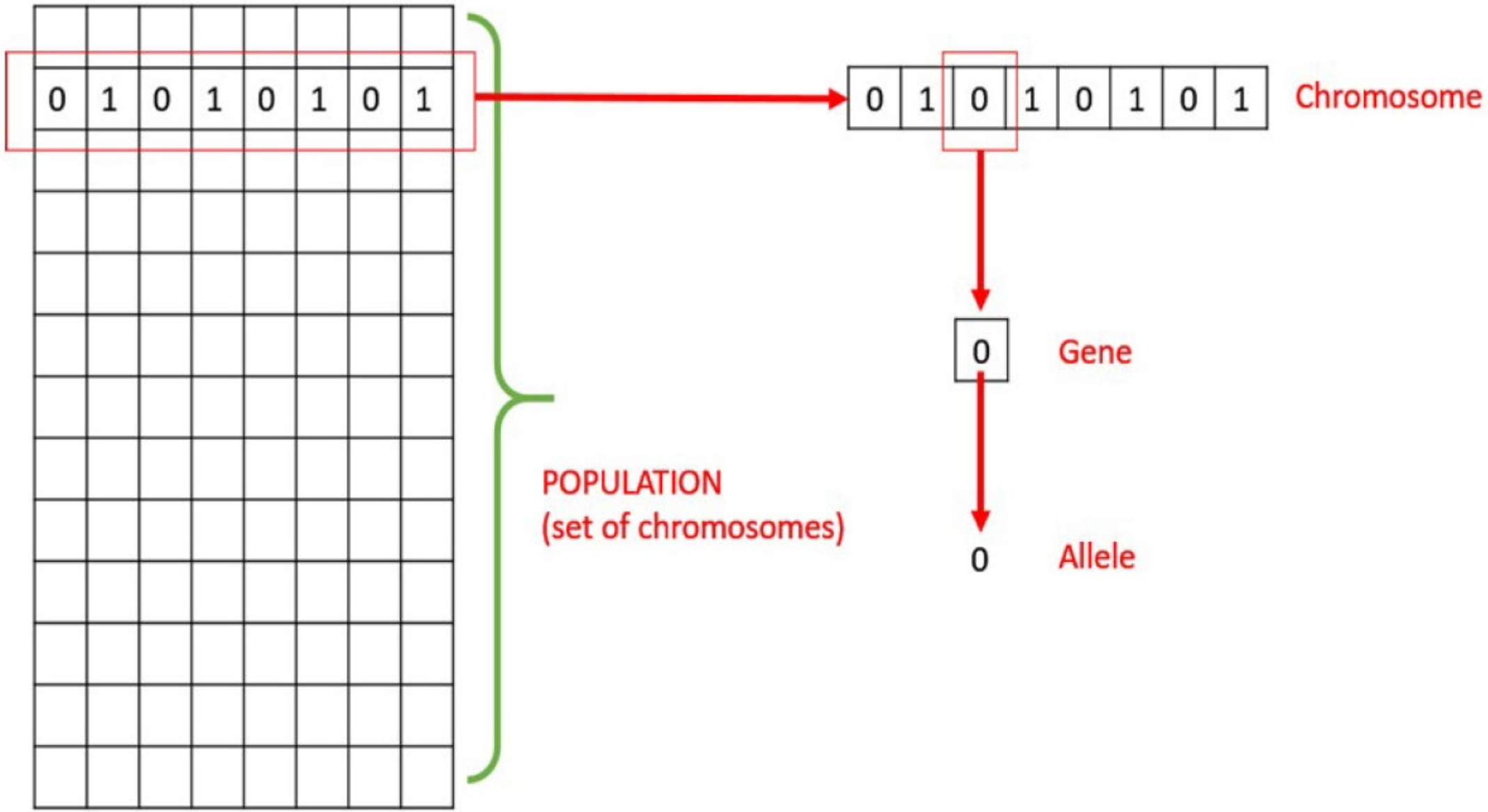
- Optimization is the process of making something better.

What are Genetic Algorithms?

- Nature has always been a great source of inspiration to all mankind. Genetic Algorithms (GAs) are search based algorithms based on the concepts of natural selection and genetics.
- GAs are a subset of a much larger branch of computation known as Evolutionary Computation.

Basic Terminology

- Population:
  - a subset of all the possible (encoded) solutions to the given problem.
- Chromosomes:
  - one such solution to the given problem.
- Gene:
  - one element position of a chromosome.
- Allele:
  - the value a gene takes for a particular chromosome.


0 1 0 1 0 1 0 1
0 1 0 1 0 1 0 1
Chromosome
0
Gene
0
Allele
POPULATION
(set of chromosomes)

Basic Terminology

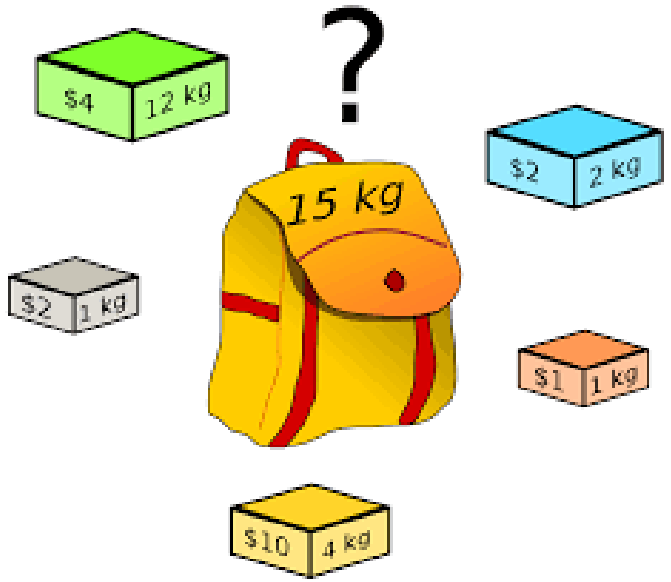
- Genotype : population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.
- Phenotype – Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.
- Decoding and Encoding – For simple problems, the phenotype and genotype spaces are the same. However, in most of the cases, the phenotype and genotype spaces are different. Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space. Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation

Basic Terminology

- Fitness Function – A fitness function simply defined is a function which takes the solution as input and produces the suitability of the solution as the output. In some cases, the fitness function and the objective function may be the same, while in others it might be different based on the problem.
- Genetic Operators – These alter the genetic composition of the offspring. These include crossover, mutation, selection, etc.

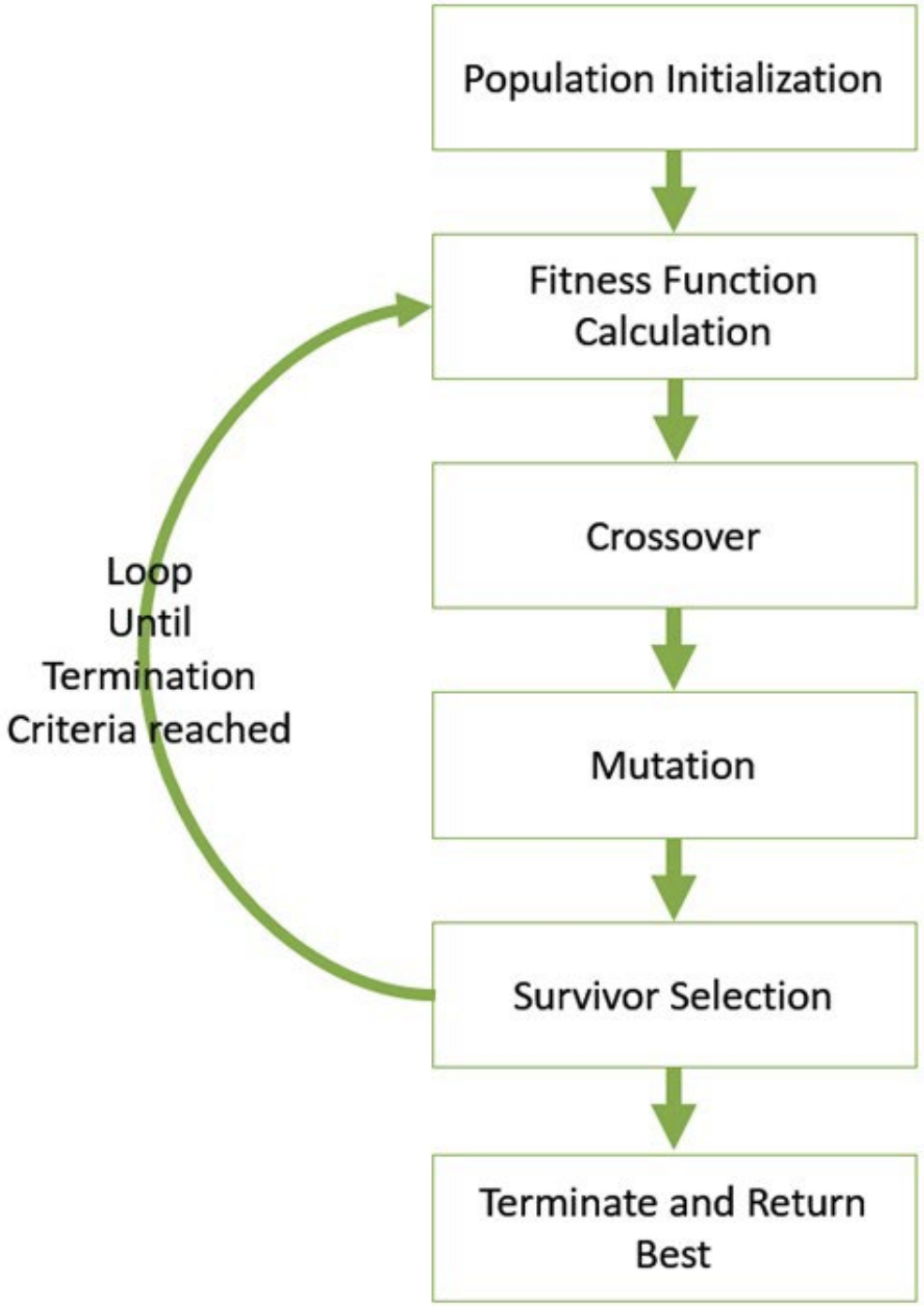
Example: Knapsack Problem

- Given a set of items,
- Each with a weight and a value,
- Determine the number of each item to include in a collection
- So that the total weight is less than or equal to a given limit
- and the total value is as large as possible.



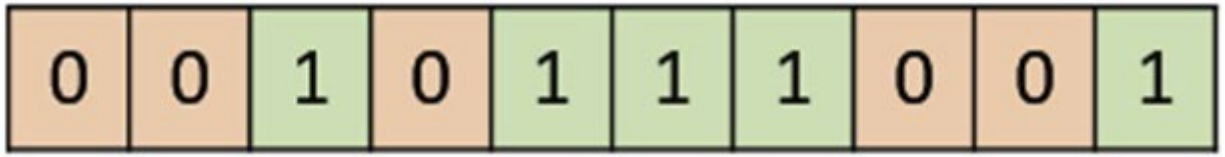
- someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items.

Basic Structure

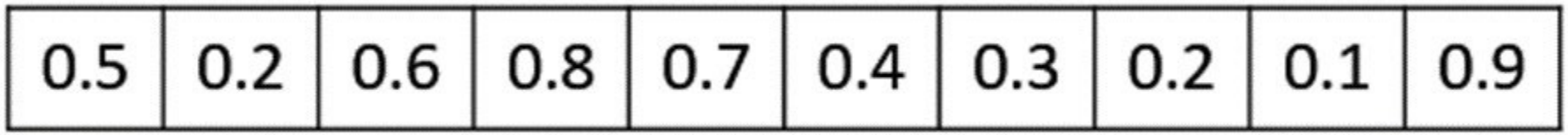


Genotype Representation

- Binary Representation

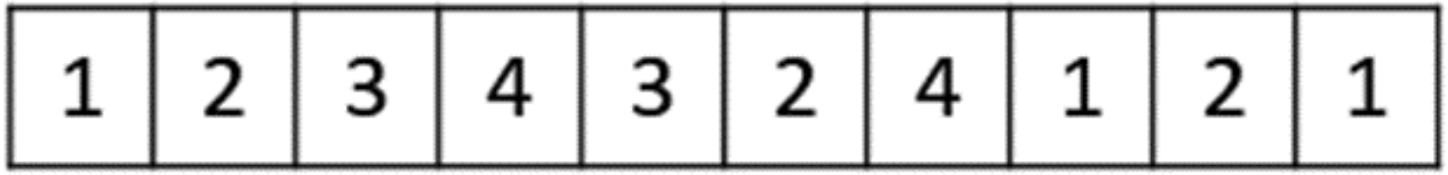


- Real-Valued Representation

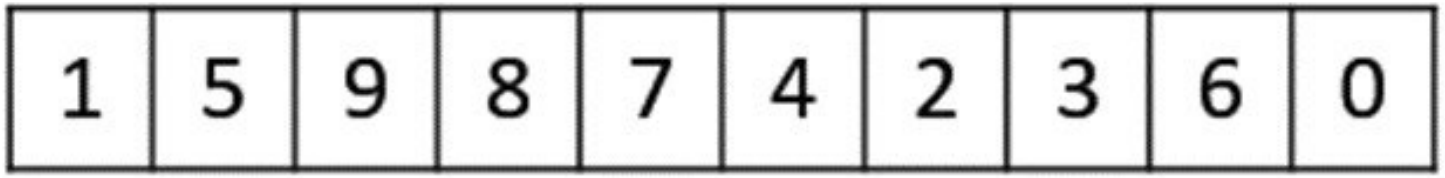


Genotype Representation

- Integer Representation



- Permutation Representation: e.x traveling salesman problem



Population

- A subset of solutions in the current generation
- Set of chromosomes

Population initialization

- Random Initialization
- Heuristic Initialization

Population model

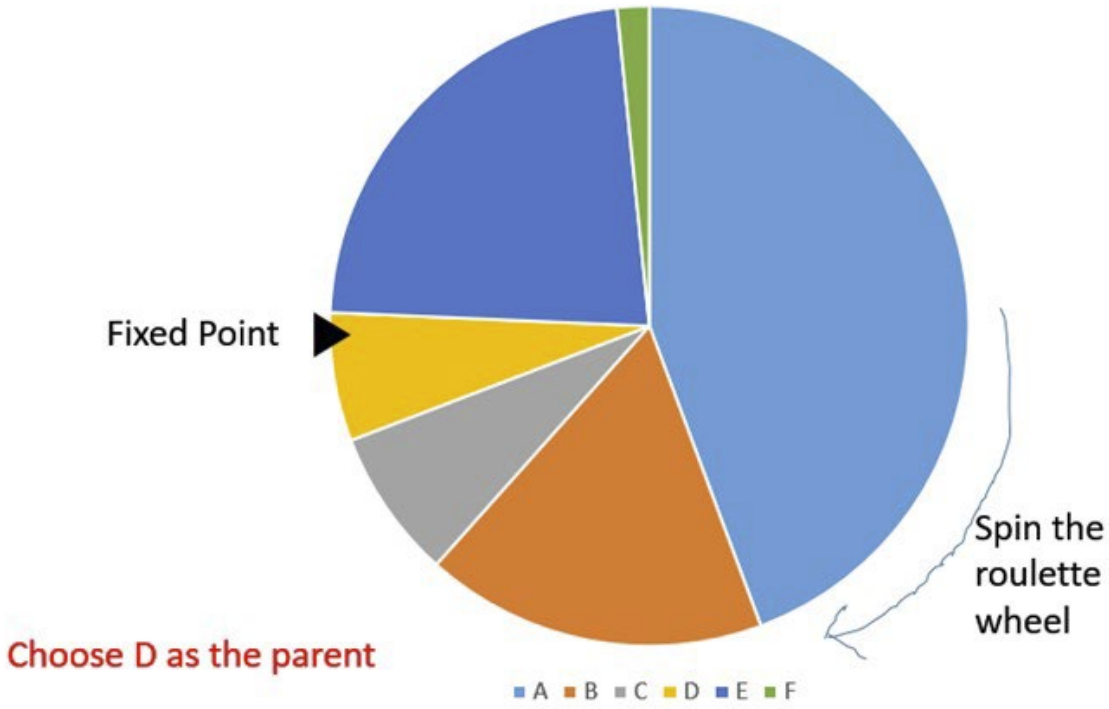
- Steady Sate
- Generational

Parent Selection

- Fitness Proportional Selection
  - Roulette Wheel Selection
  - Stochastic Universal Sampling (SUS)
  - Tournament Selection
  - Rank Selection

Parent Selection

- Roulette Wheel Selection

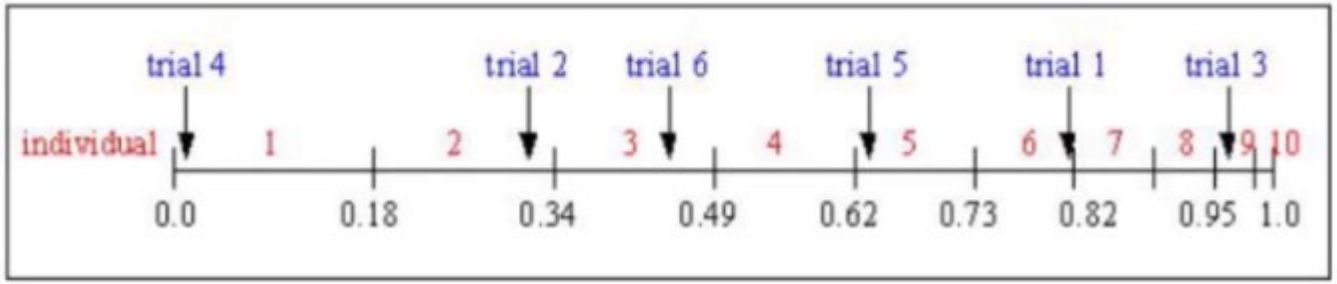


Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Roulette Wheel Selection

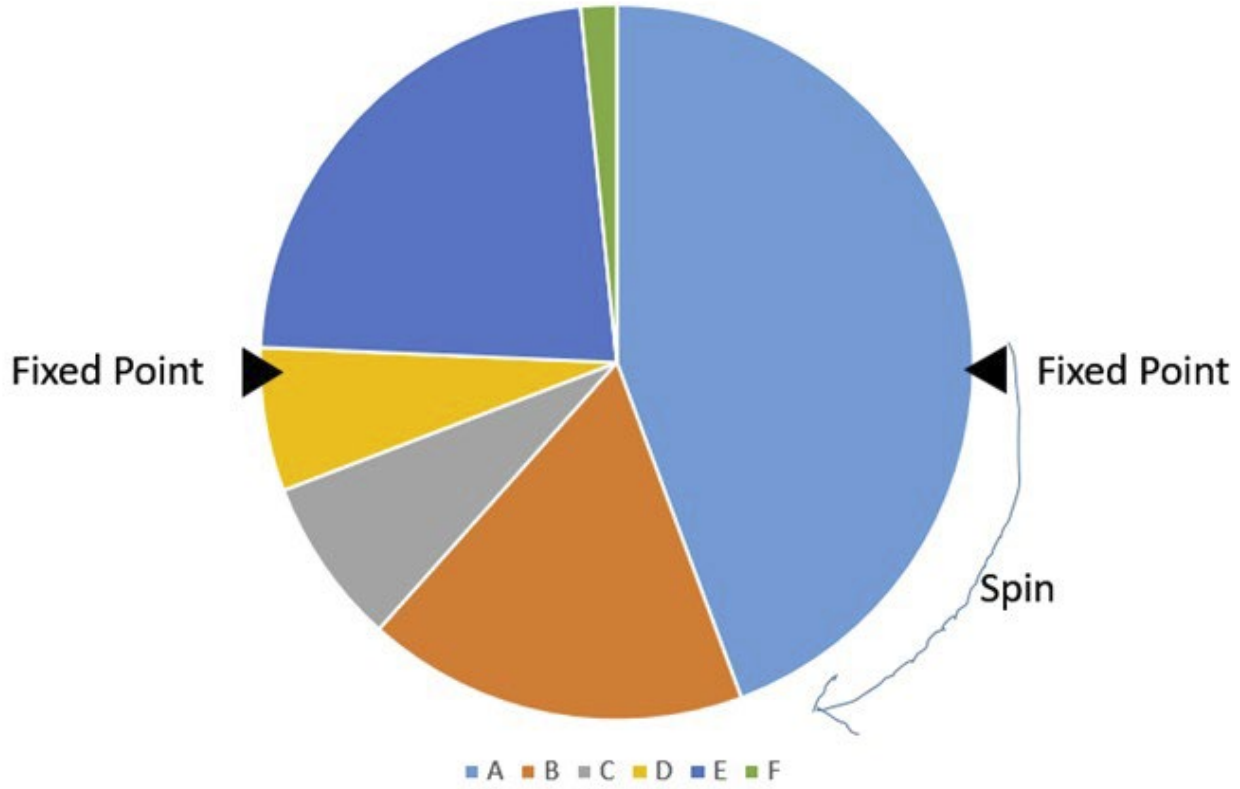
# of Individual	1	2	3	4	5	6	7	8	9	10	11
Rank	11	10	9	8	7	6	5	4	3	2	1
Fitness value	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.0
Selection Probability	0.18 =2/11	0.16=1. 8/11	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02	0.0

0.81	0.32	0.96	0.01	0.65	0.42



Parent Selection

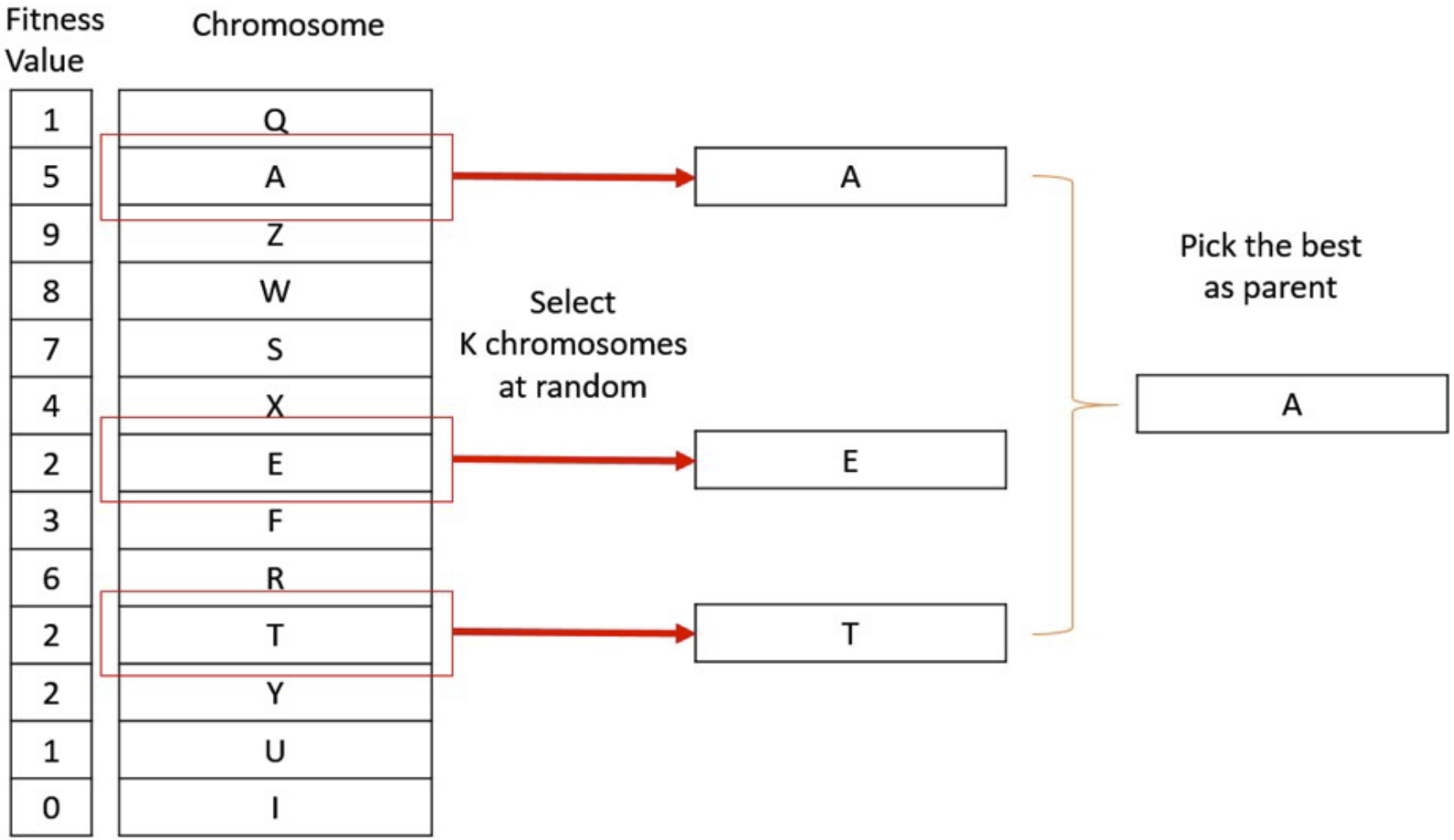
- Stochastic Universal Sampling (SUS)



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Parent Selection

- Tournament Selection



Parent Selection

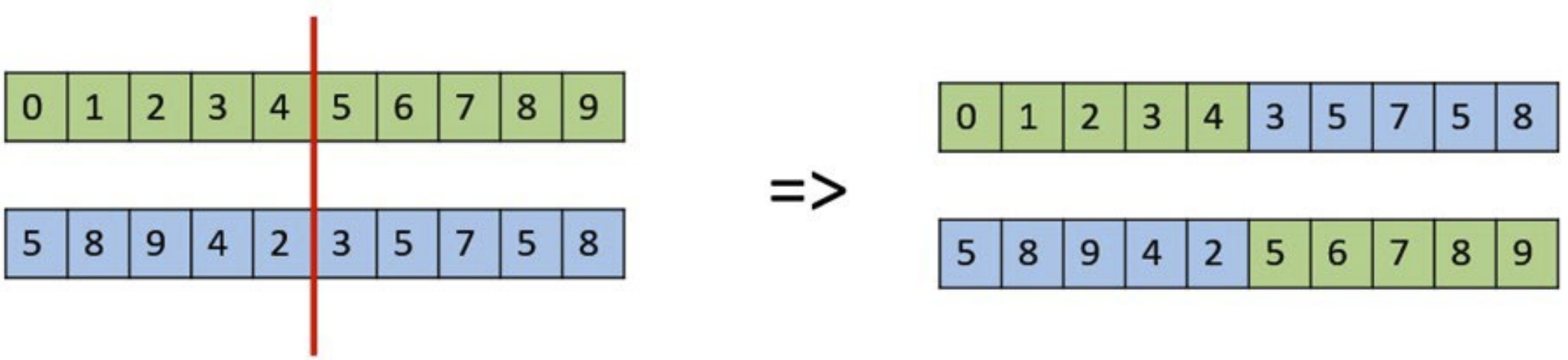
- Rank Selection:
  - rank is assigned based on the fitness, then choose proportional to rank

Crossover

- Two parents are selected and one or more offsprings are produced using the genetic material of the parents
  - One point crossover
  - Multi Point Crossover
  - Uniform Crossover

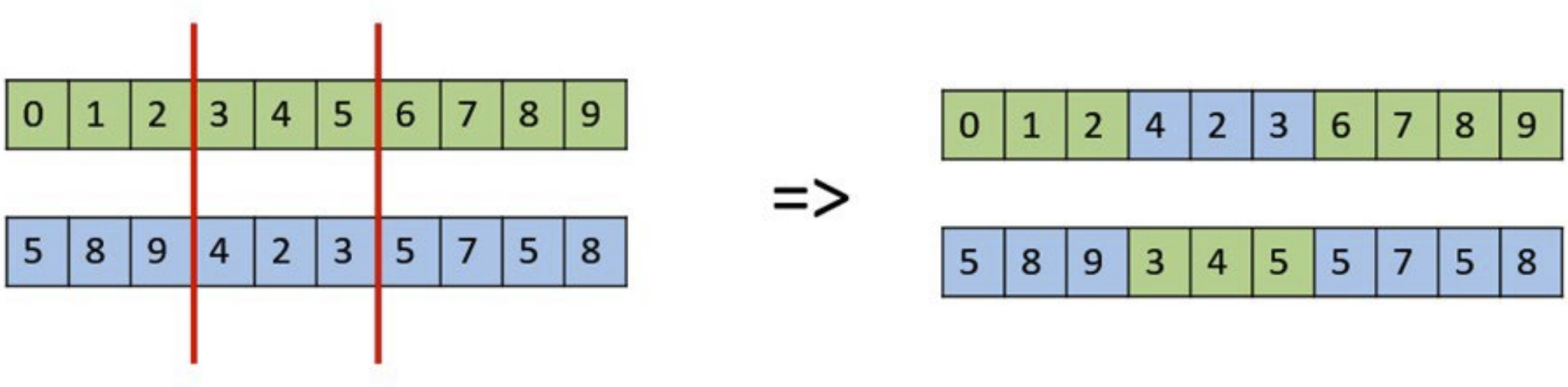
Crossover

- One point crossover



Crossover

- Multi Point Crossover

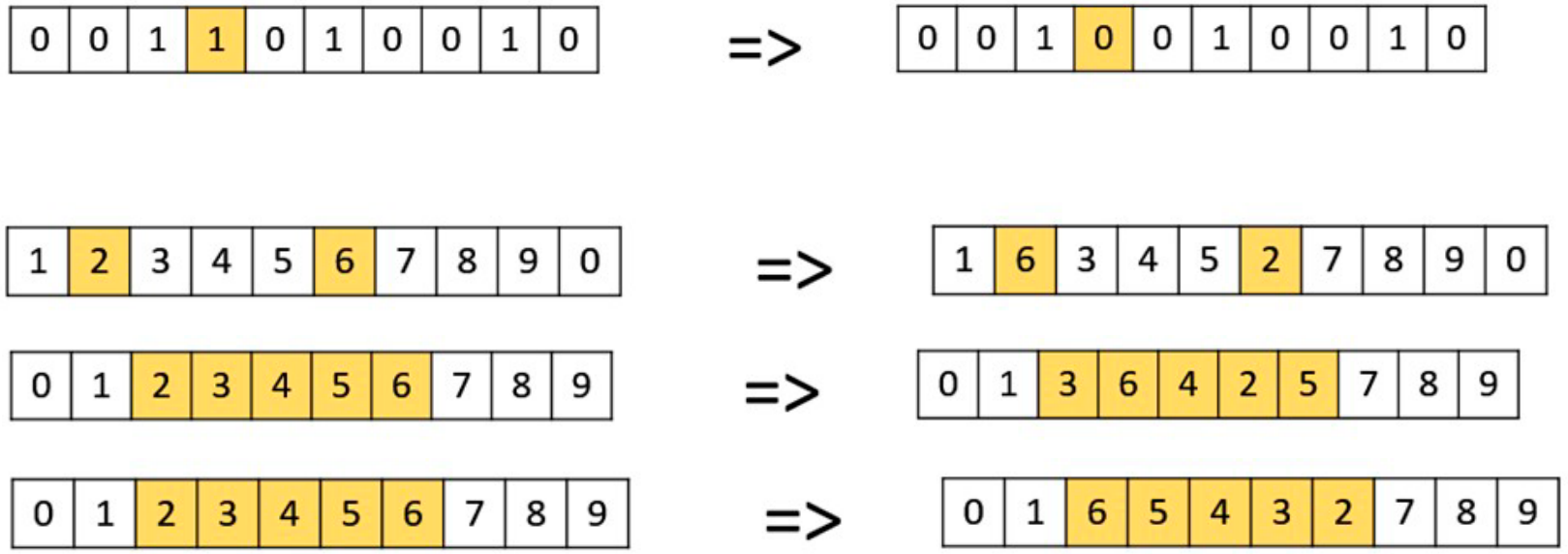


Crossover

- Uniform Crossover

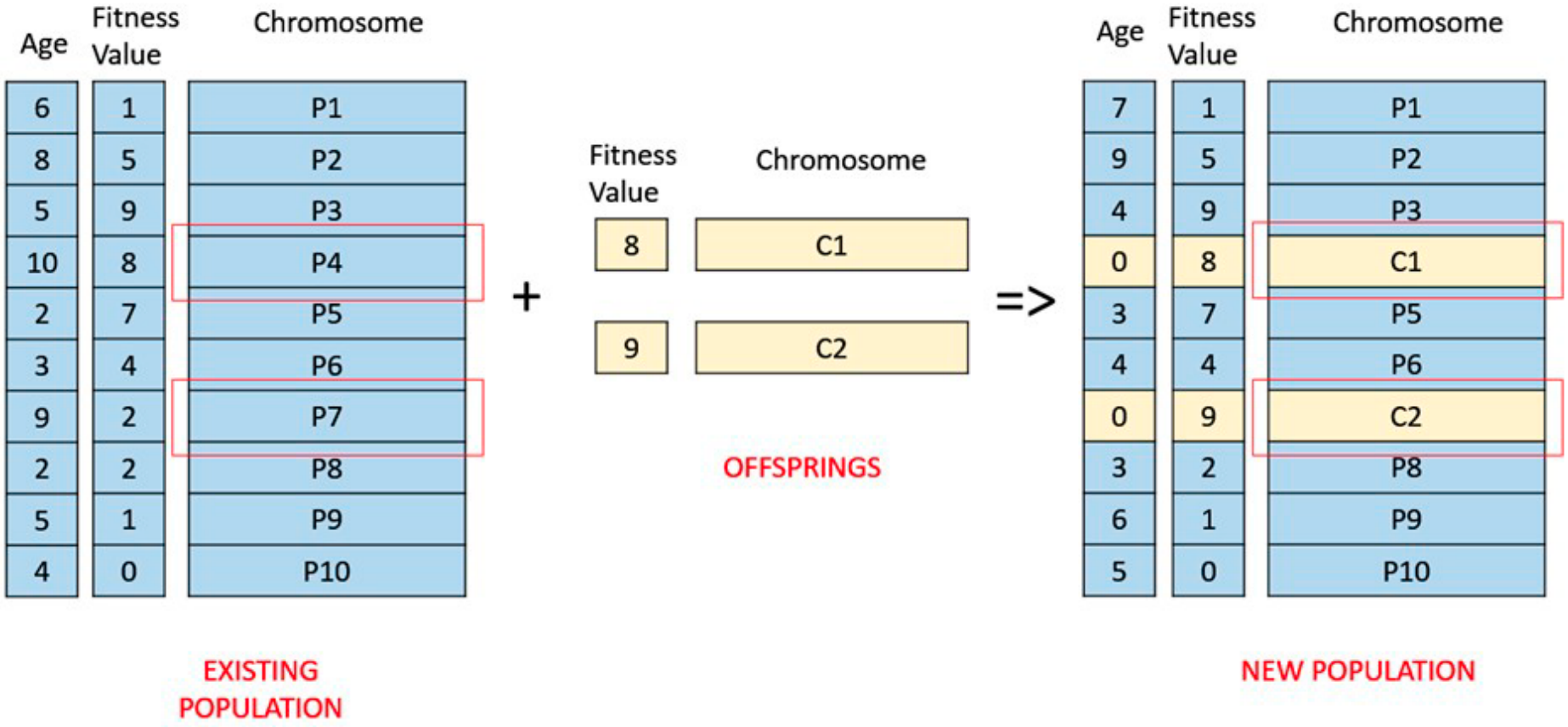
Mutation

- used to maintain and introduce diversity in the genetic population
- Mutation Operators:
  - Bit Flip Mutation
  - Random Resetting
  - Swap Mutation
  - Scramble Mutation
  - Inversion Mutation



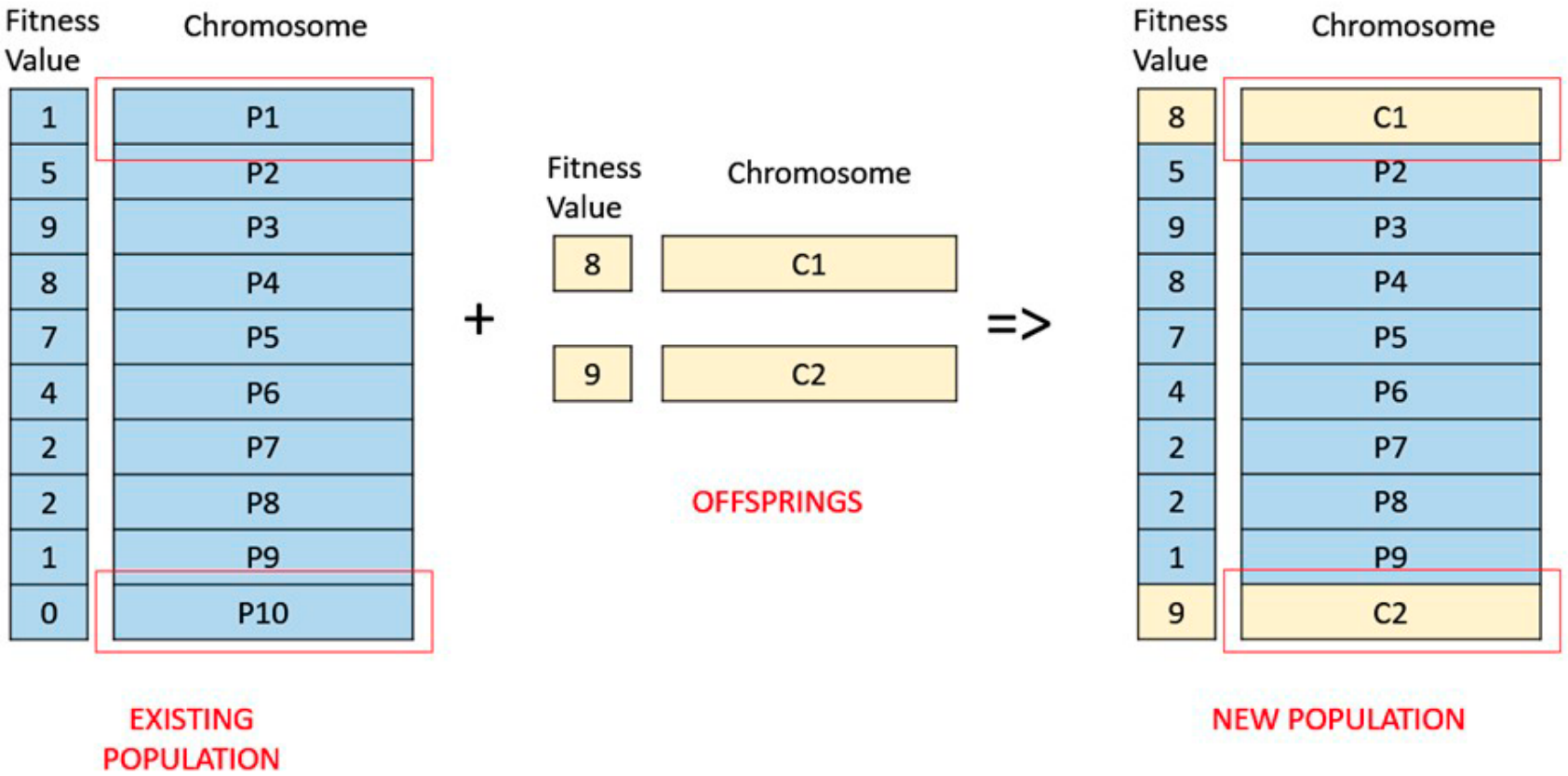
Survivor Selection

- Age Based Selection



Survivor Selection

- Fitness Based Selection



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

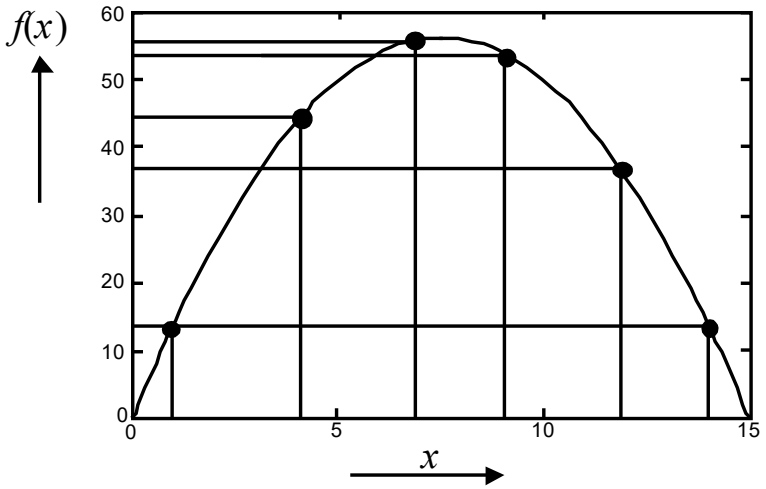
Genetic algorithms: case study

- 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

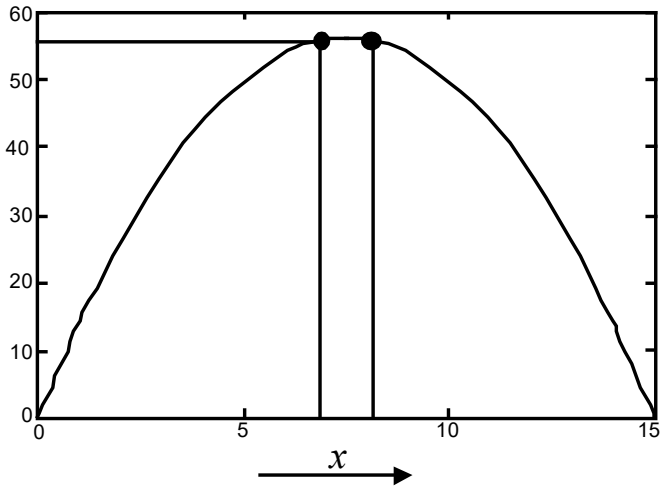
$$f(x) = 15\,x - x^2$$

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.



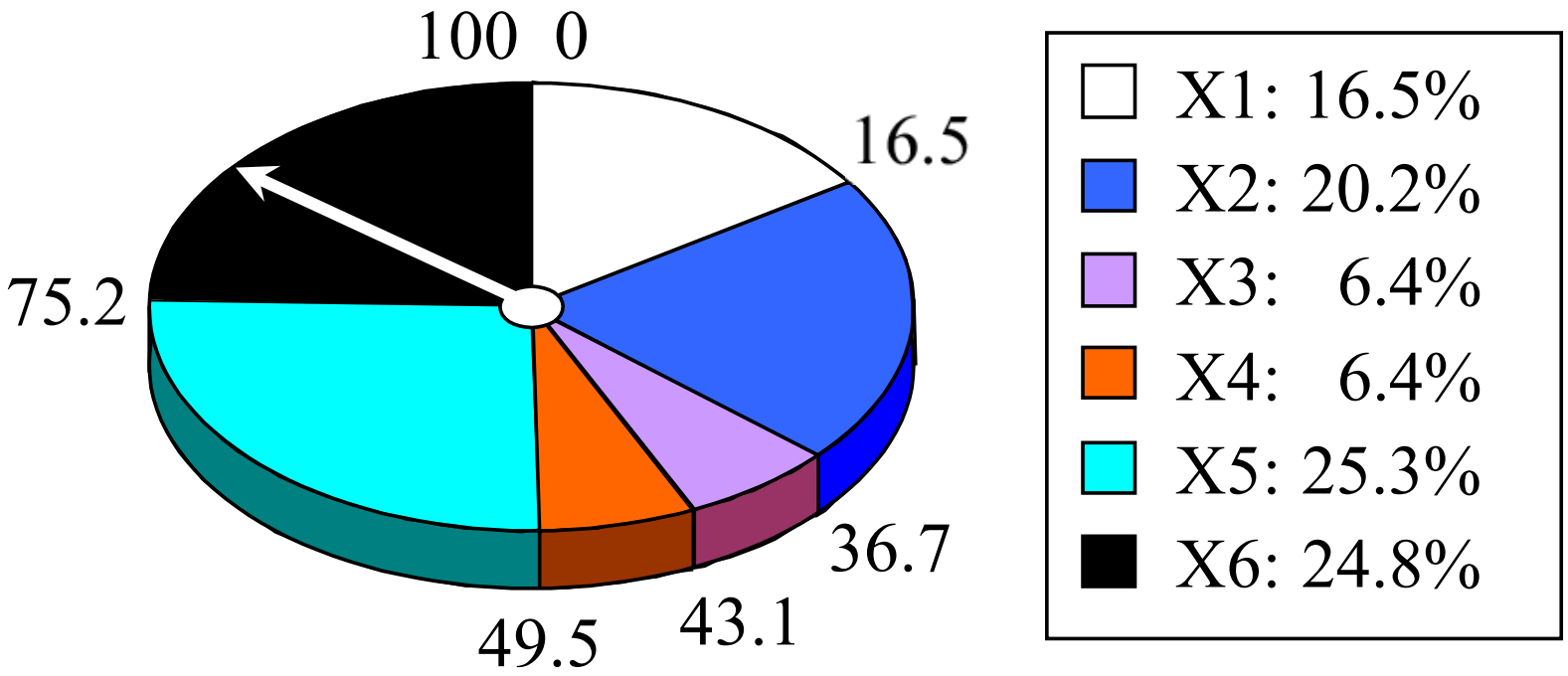
(*b*) Chromosome final locations.

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.



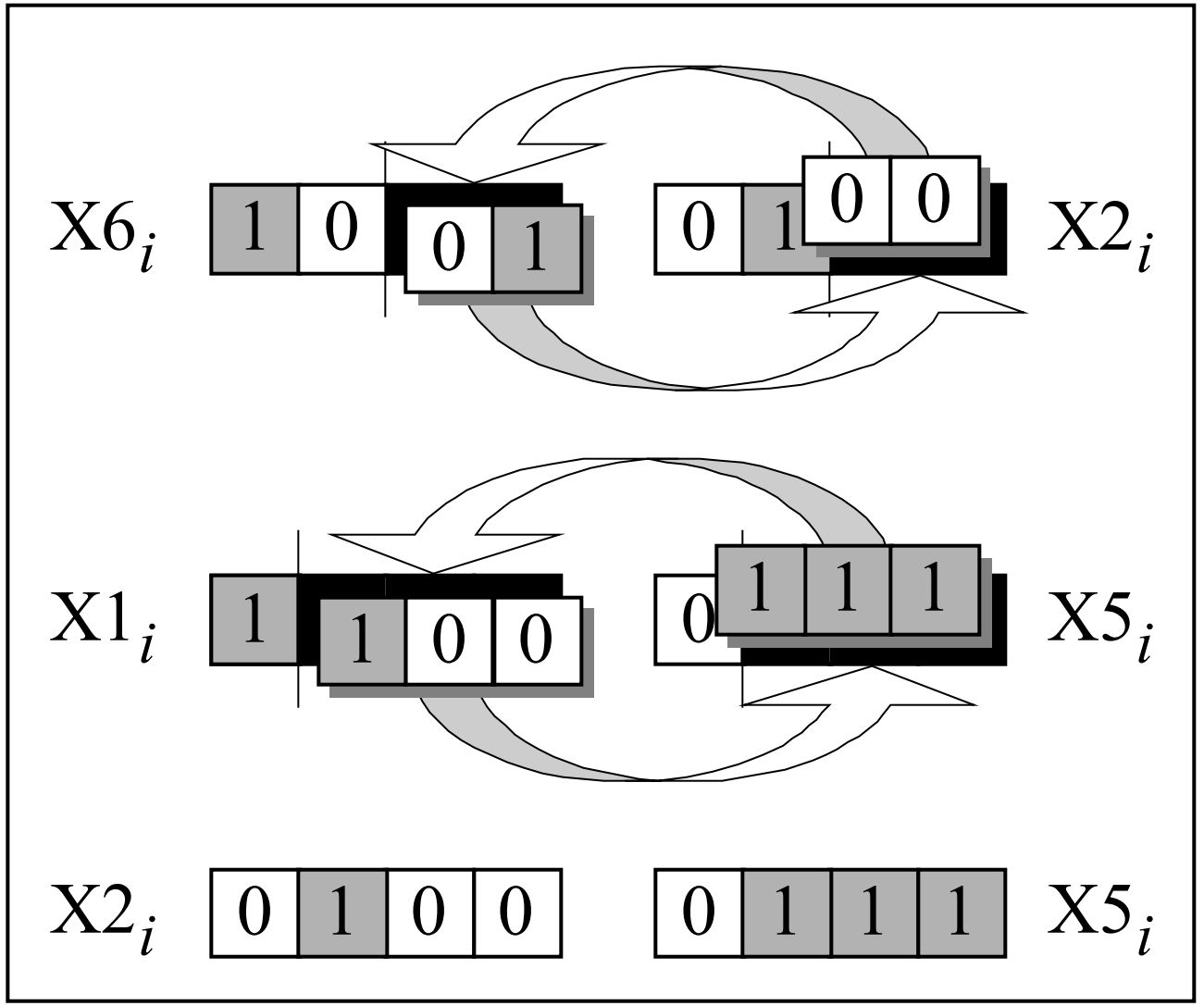
Crossover operator

- 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.

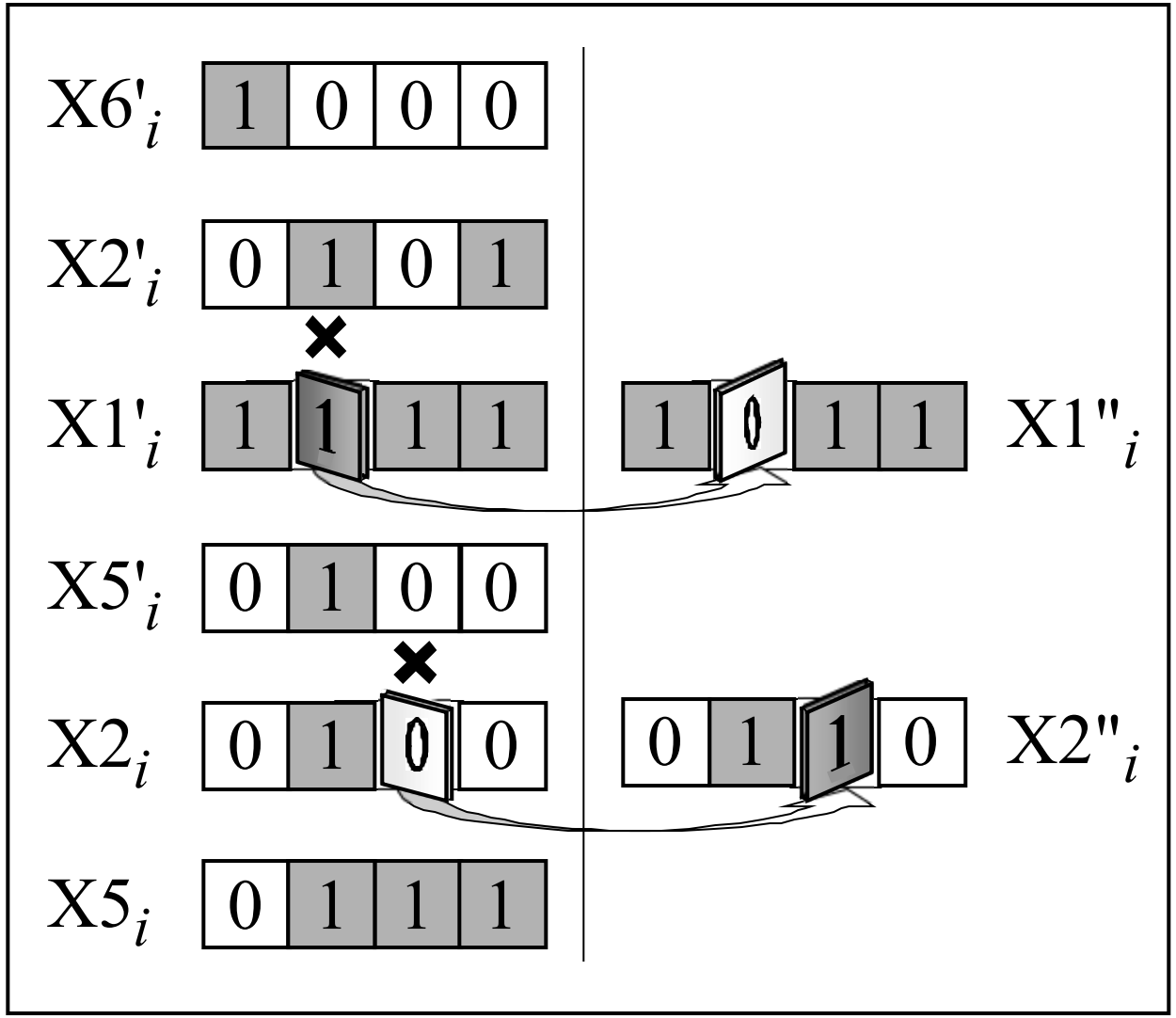
Crossover operator

- 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.

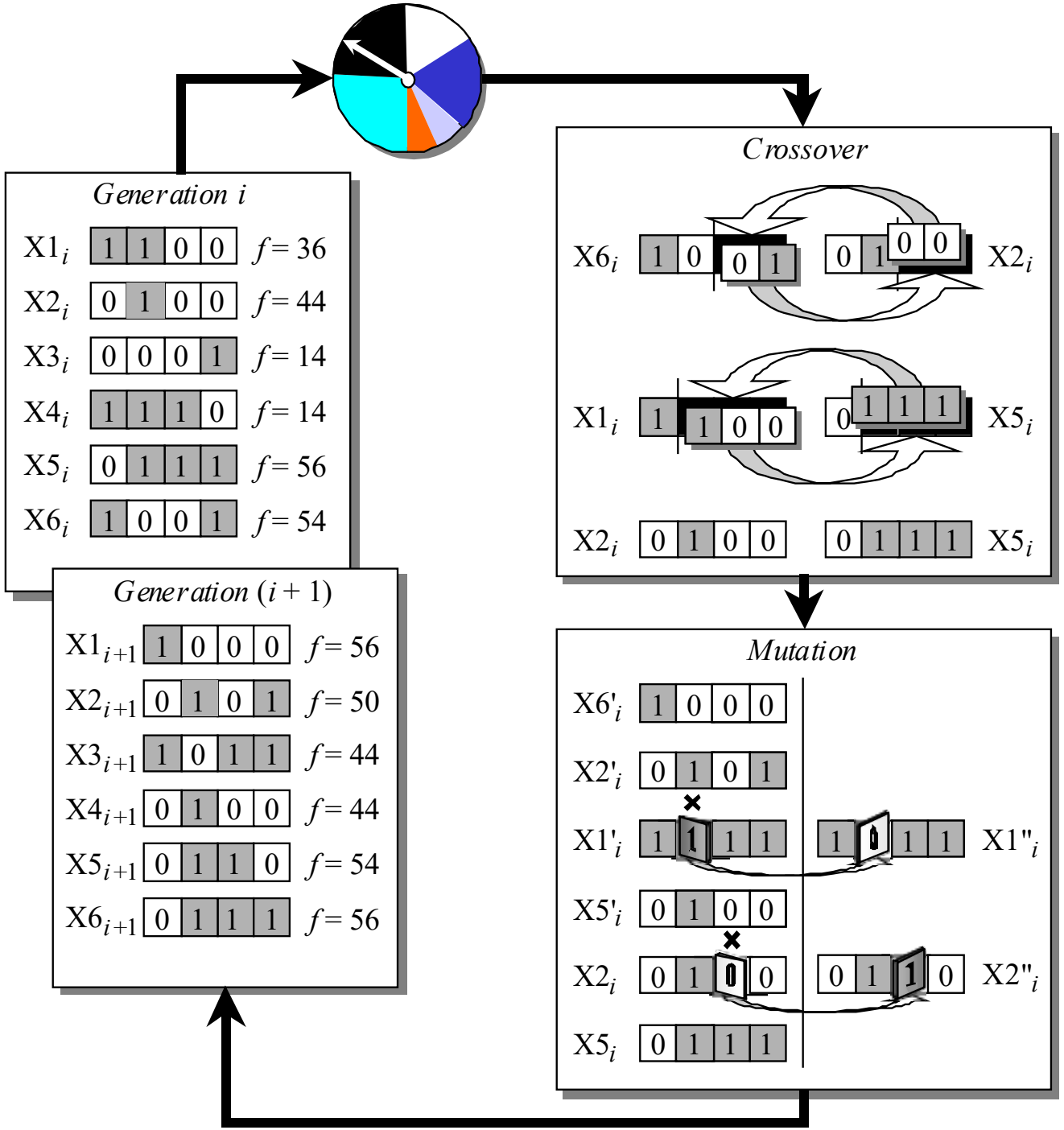
Crossover



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

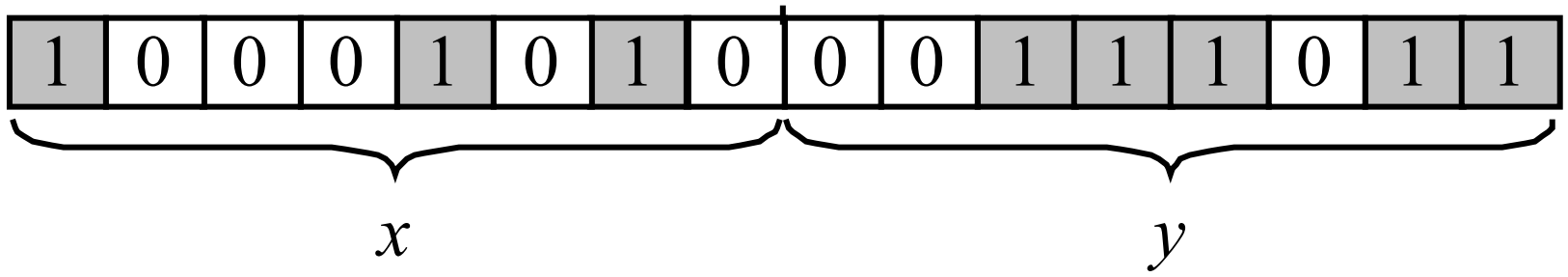
- Suppose it is desired to find the maximum of the “peak” function of two variables:

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

- 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:



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:

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:

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

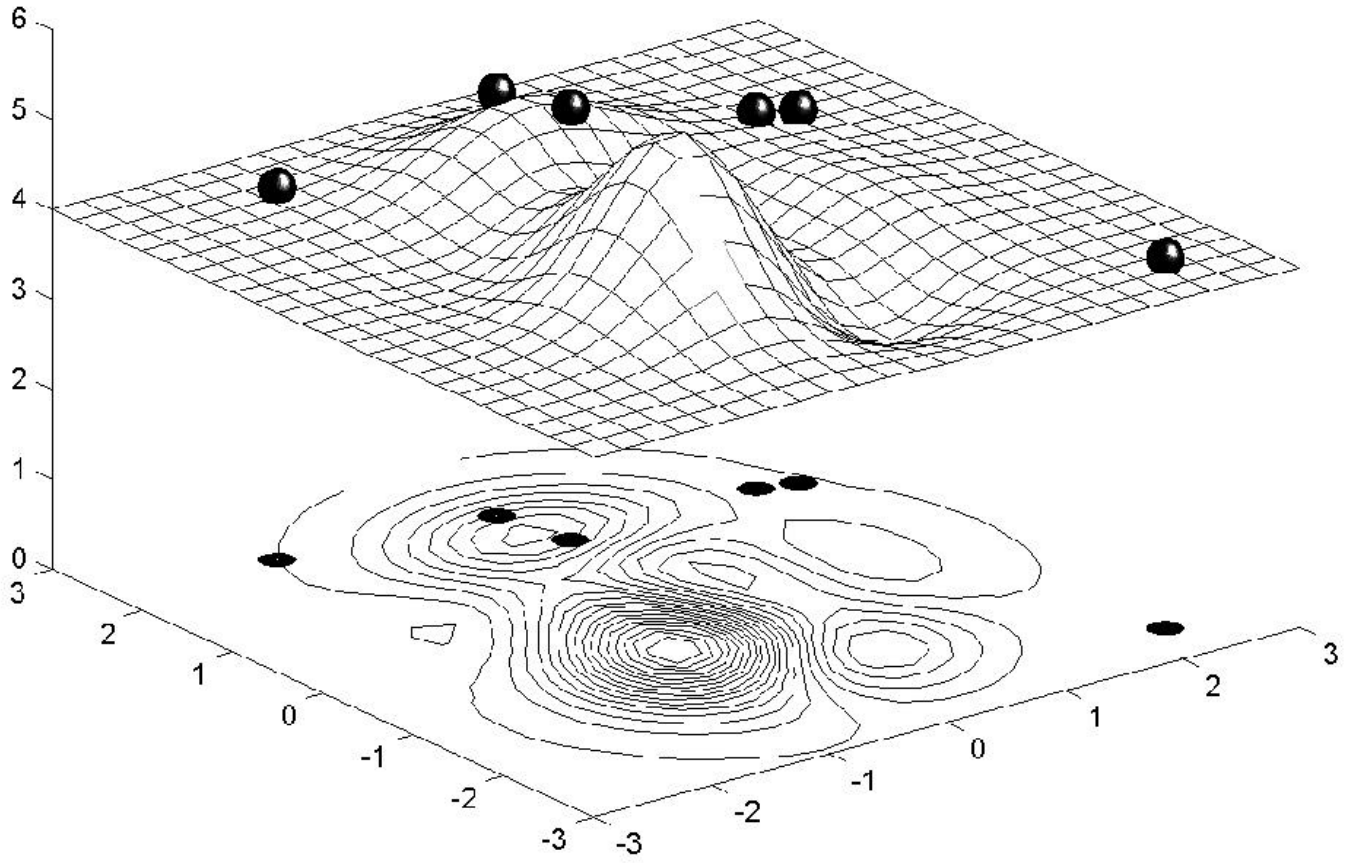
and

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

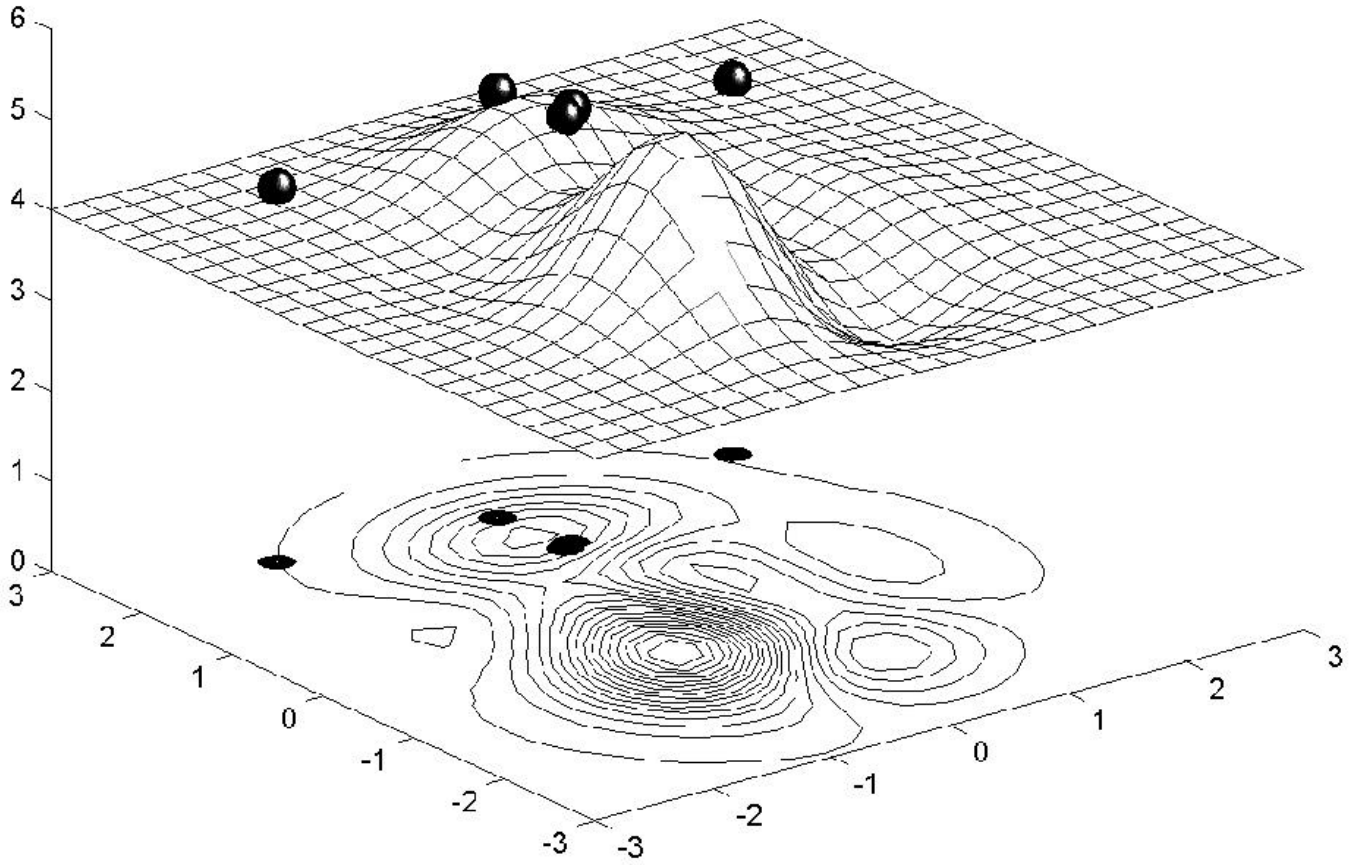
Genetic algorithms: case study

- 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.

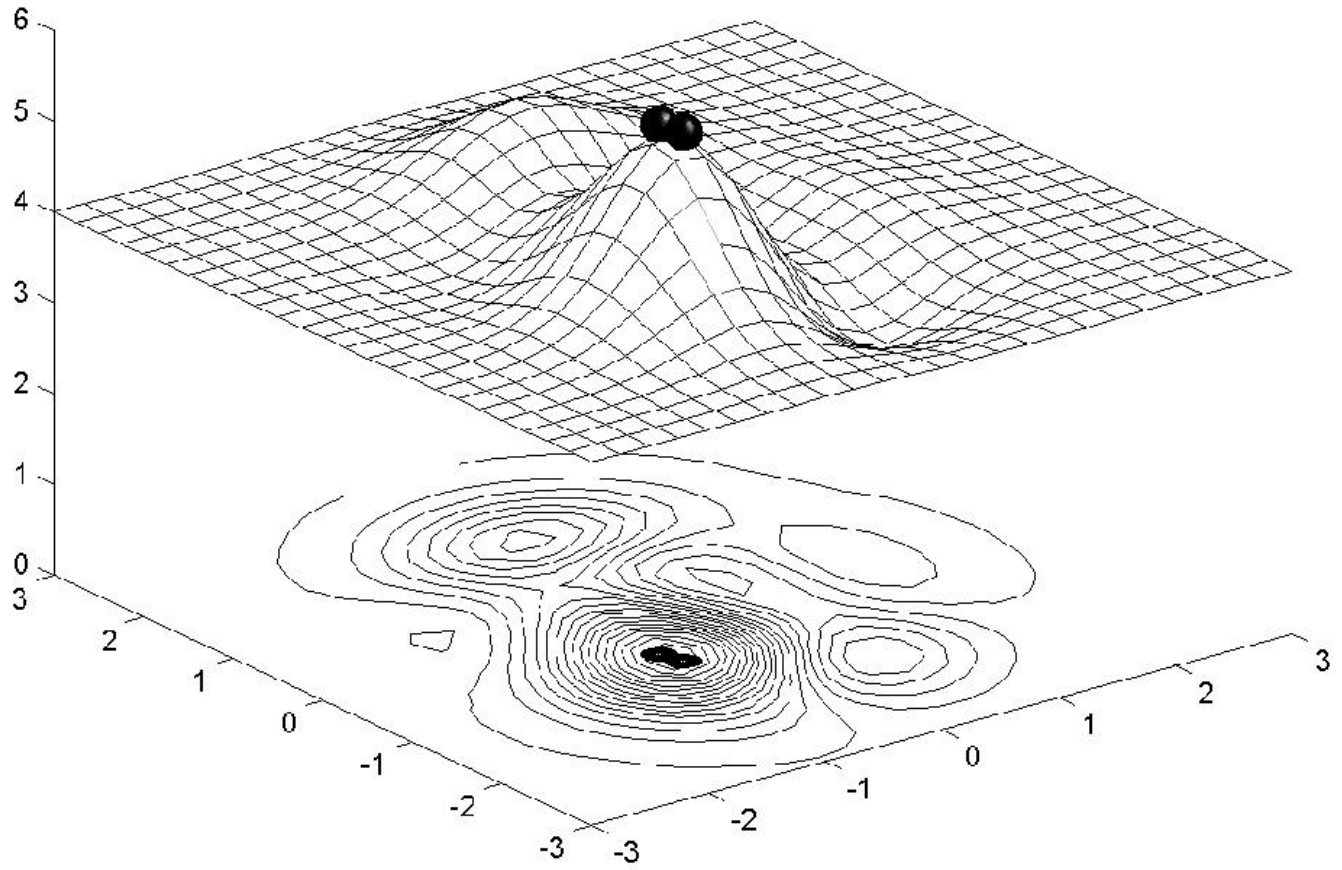
Chromosome locations on the surface of the “peak” function: initial population



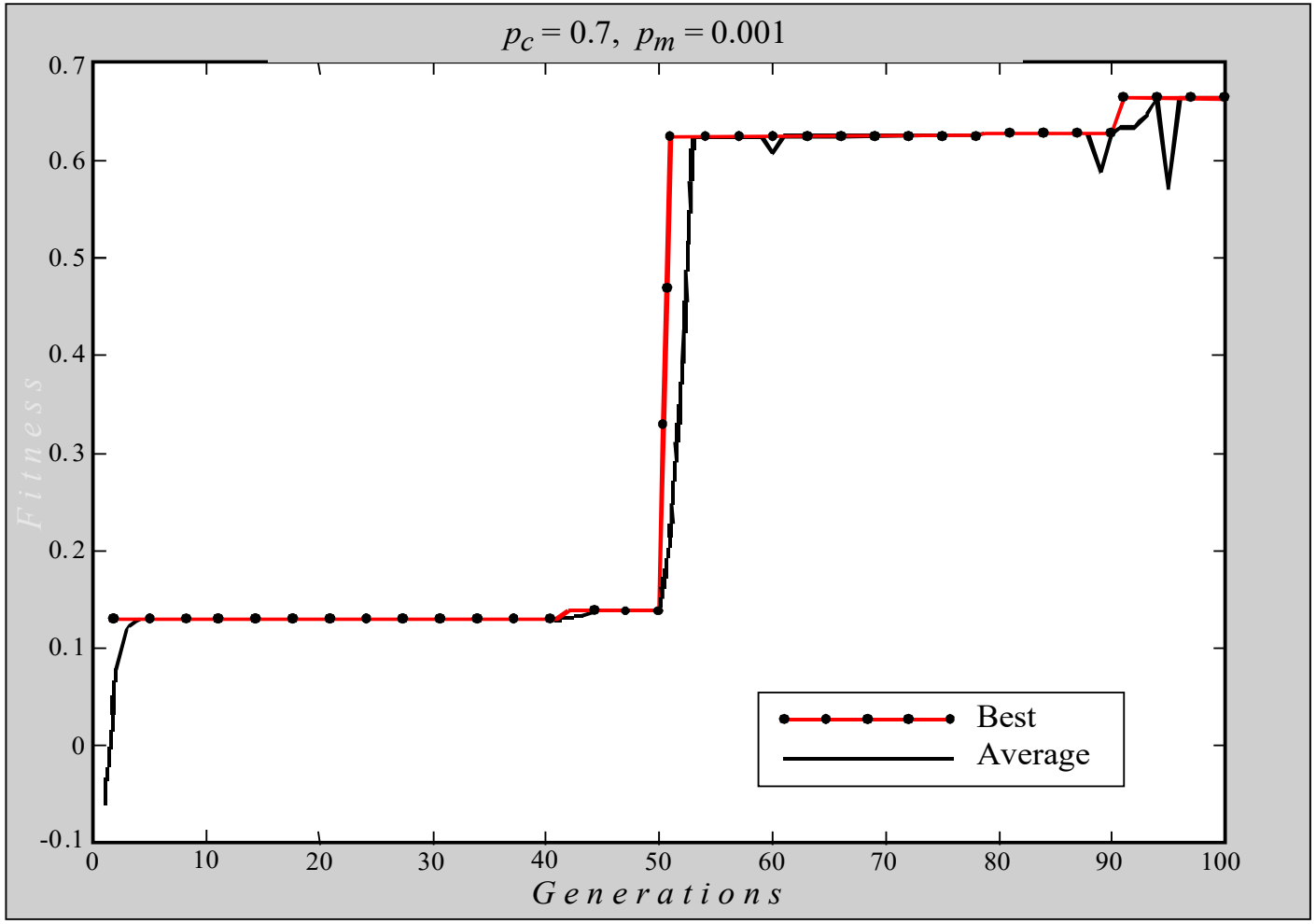
Chromosome locations on the surface of the “peak” function: initial population



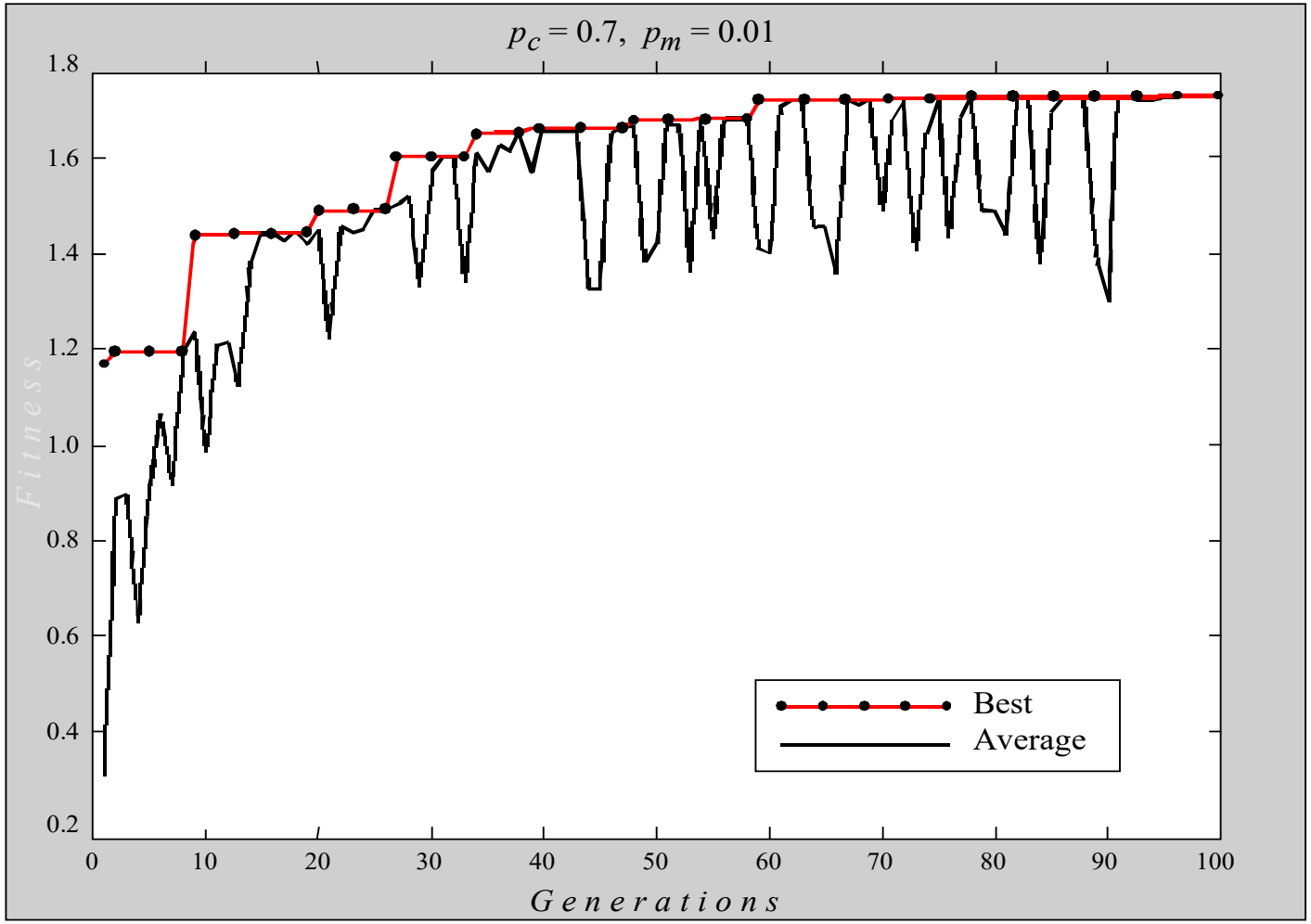
Chromosome locations on the surface of the “peak” function: initial population



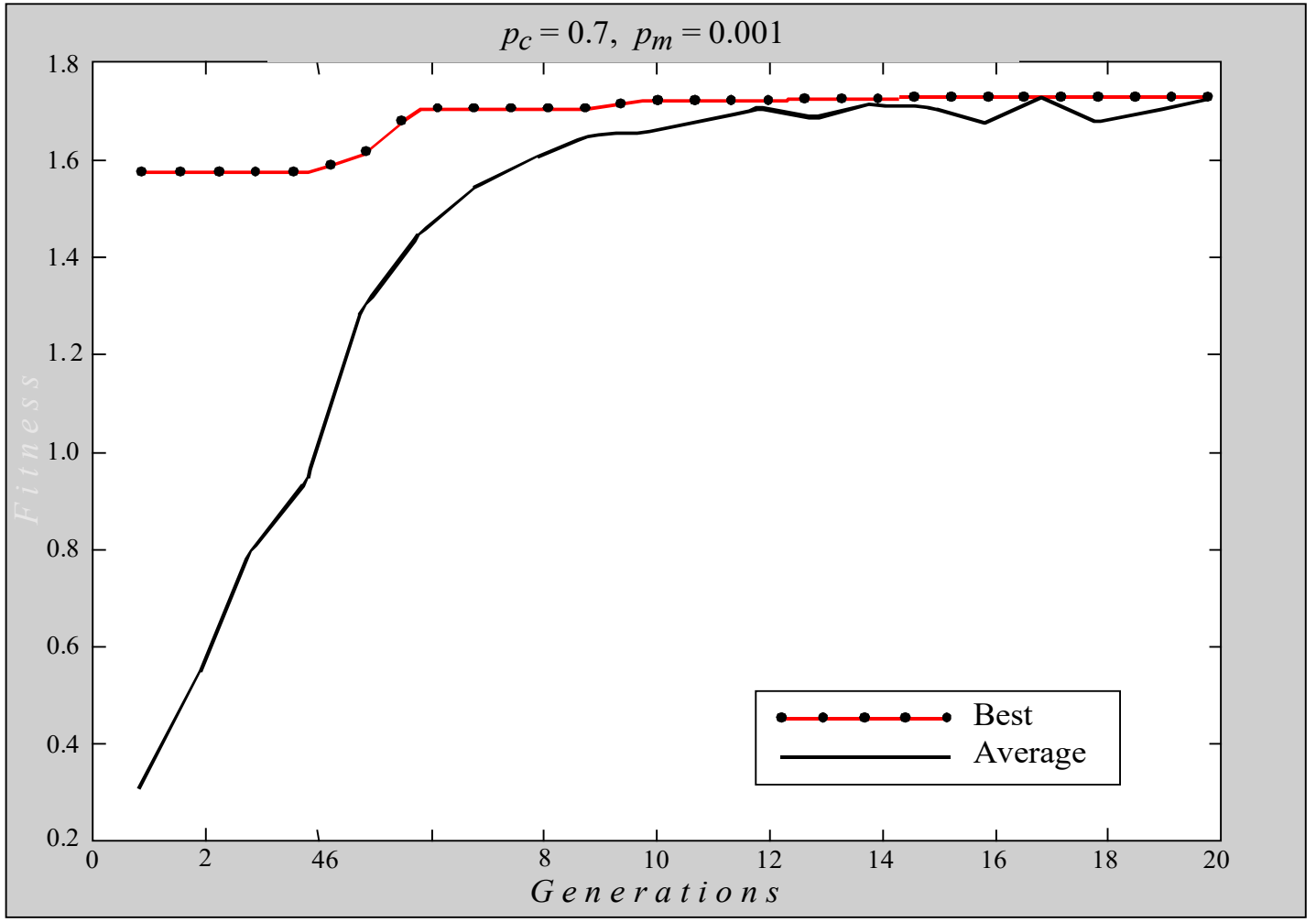
Performance graphs for 100 generations of 6 chromosomes: Local maximum



Performance graphs for 100 generations of 6 chromosomes: Global maximum



Performance graphs for 20 generations of 60 chromosomes

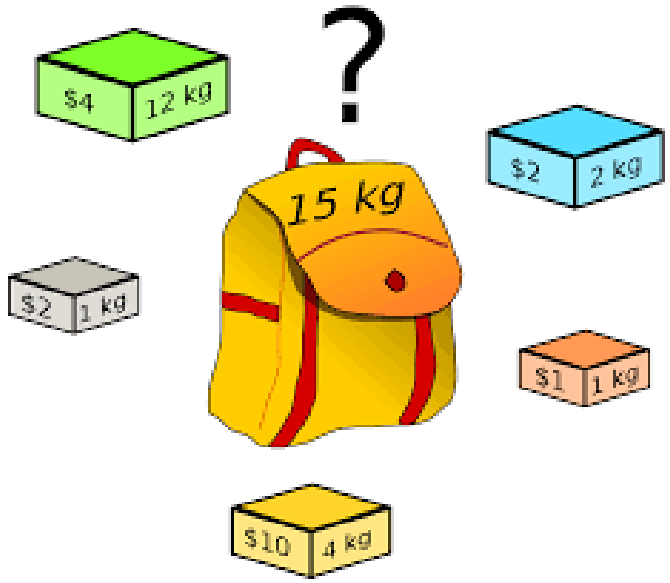


Steps in the GA development

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.

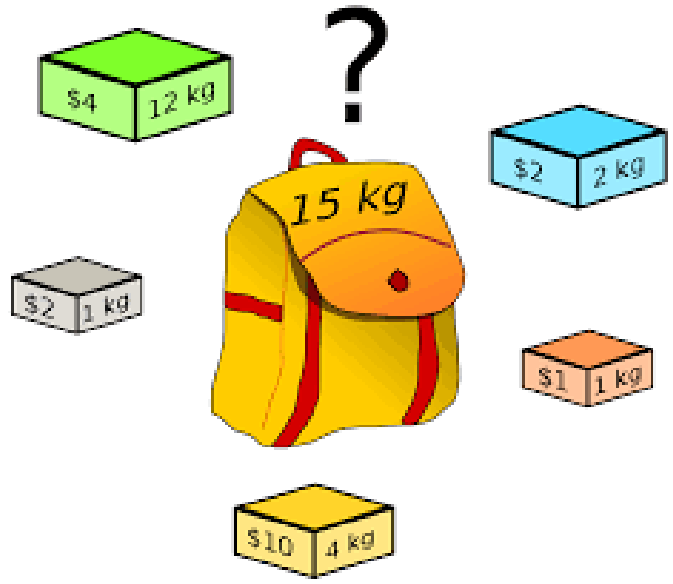
Case study: Knapsack Problem

- 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

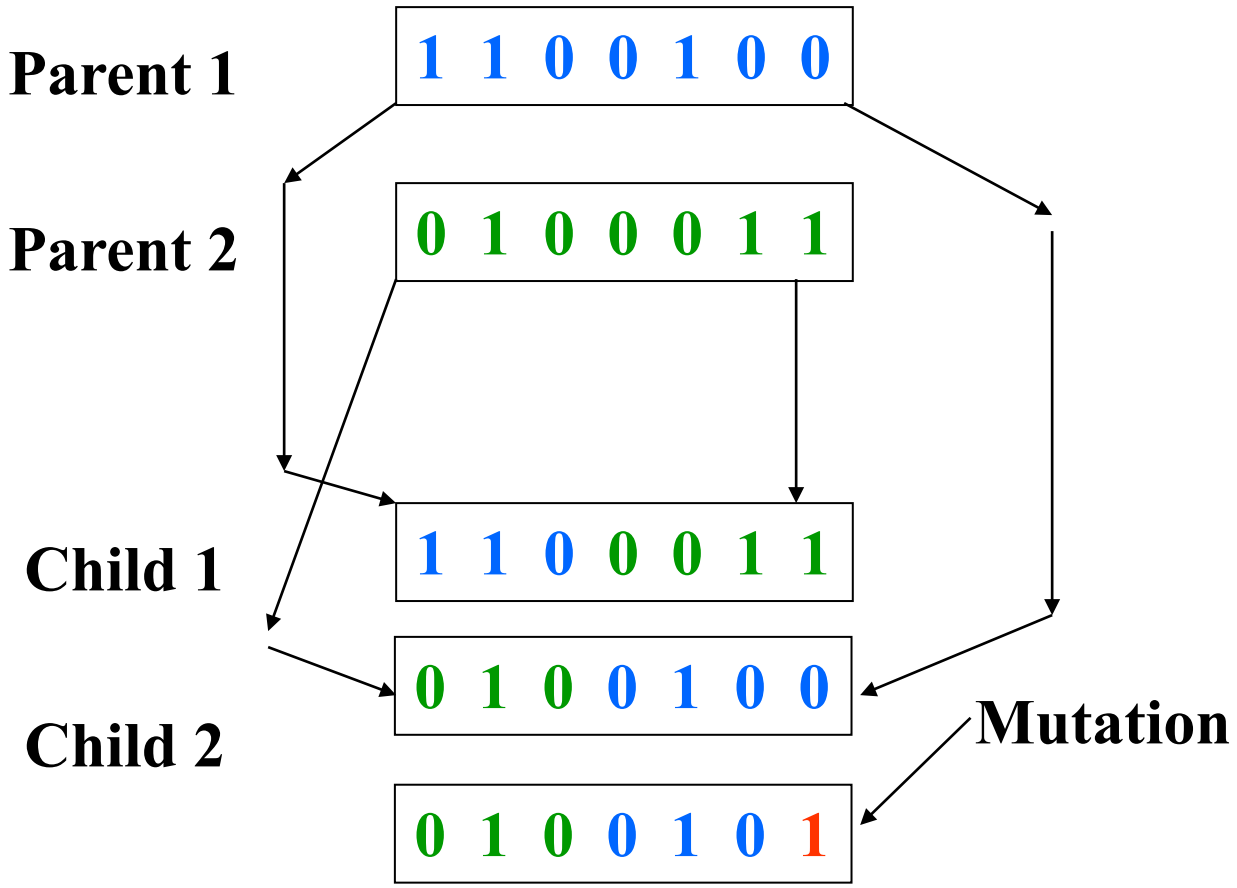
Item	1	2	3	4	5	6	7
Chro	0	1	0	1	0	1	0
Benefit	5	8	3	2	7	9	4
Weight	7	8	4	10	4	6	4

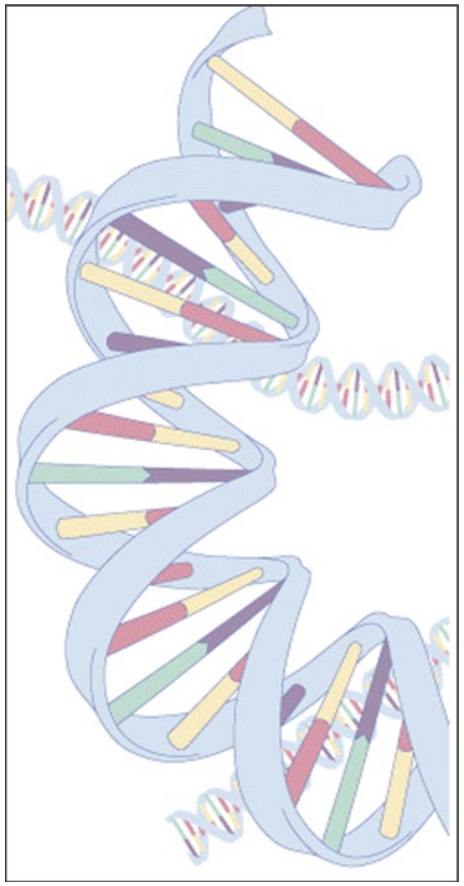
b) 1100100: Benefit= 20, Weight= 19.

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

=> We select Chromosomes b & c.

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





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].

Case study: maintenance scheduling

- Maintenance scheduling problems are usually solved using a combination of search techniques and heuristics.
- These problems are complex and difficult to solve.
- They are NP-complete and cannot be solved by combinatorial search techniques.
- Scheduling involves competition for limited resources, and is complicated by a great number of badly formalized constraints.

Case study:
Scheduling of 7 units in 4 equal intervals

- The problem constraints:
- The maximum loads expected during four intervals are 80, 90, 65 and 70 MW;
- Maintenance of any unit starts at the beginning of an interval and finishes at the end of the same or adjacent interval.
- The maintenance cannot be aborted or finished earlier than scheduled;
- The net reserve of the power system must be greater or equal to zero at any interval.

Case study:
Scheduling of 7 units in 4 equal intervals

- The problem constraints:
  - The maximum loads expected during four intervals are 80, 90, 65 and 70 MW;
  - Maintenance of any unit starts at the beginning of an interval and finishes at the end of the same or adjacent interval.
  - The maintenance cannot be aborted or finished earlier than scheduled;
  - The net reserve of the power system must be greater or equal to zero at any interval.
- The optimum criterion is the maximum of the net reserve at any maintenance period.

Case study:
Scheduling of 7 units in 4 equal intervals

Unit number	*Unit capacity, MW*	*Number of intervals required for unit maintenance*
12	0	2
21	5	2
33	5	1
44	0	1
51	5	1
61	5	1
71	0	1

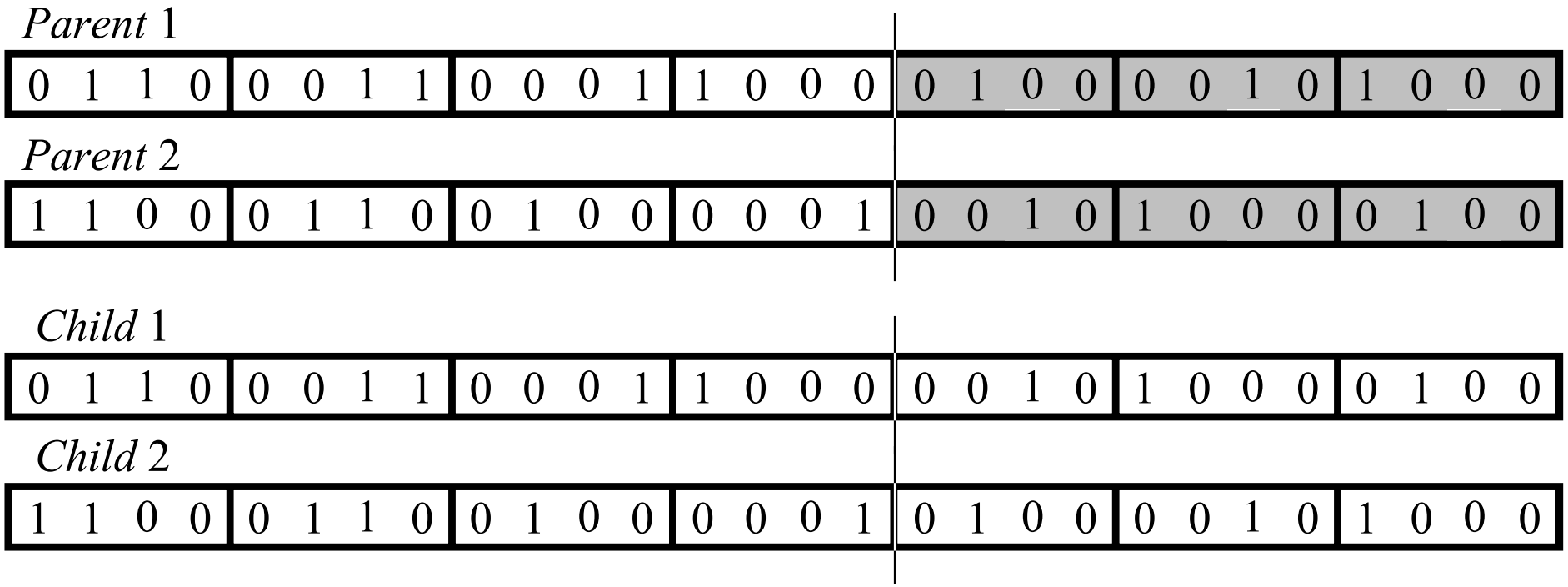
Case study: Unit gene pools

Unit 1:	1 1 0 0	0 1 1 0	0 0 1 1	
Unit 2:	1 1 0 0	0 1 1 0	0 0 1 1	
Unit 3:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 4:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 5:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 6:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 7:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1

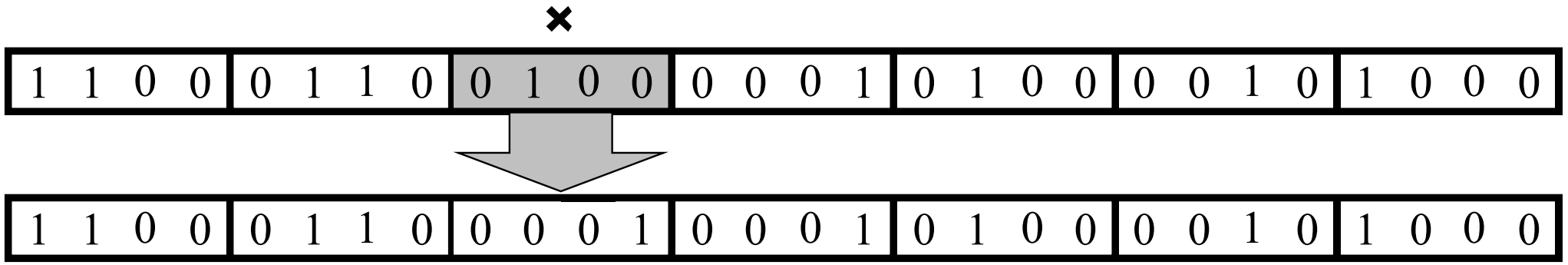
Chromosome for the scheduling problem

Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6	Unit 7
0 1 1 0	0 0 1 1	0 0 0 1	1 0 0 0	0 1 0 0	0 0 1 0	1 0 0 0

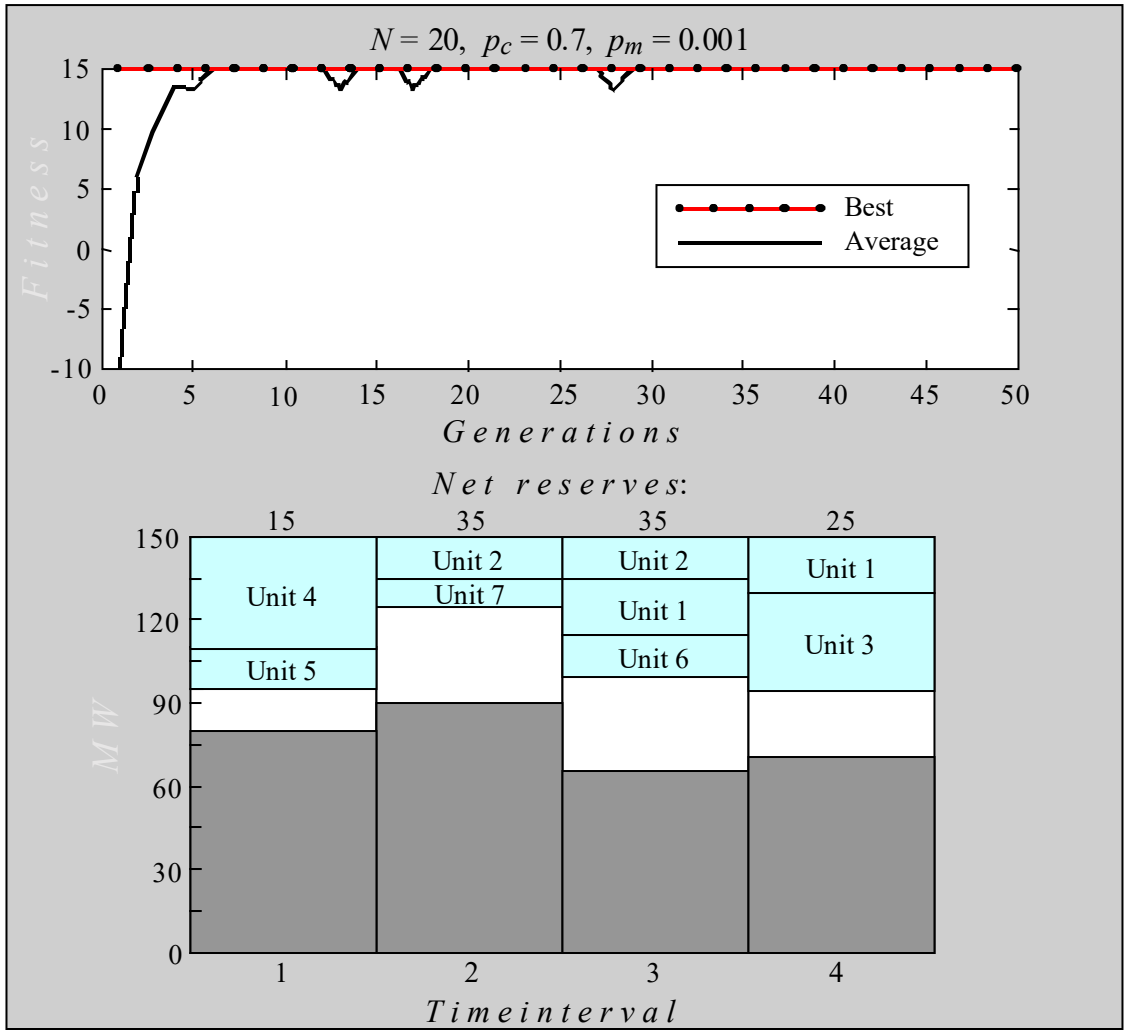
Case study: The crossover operator



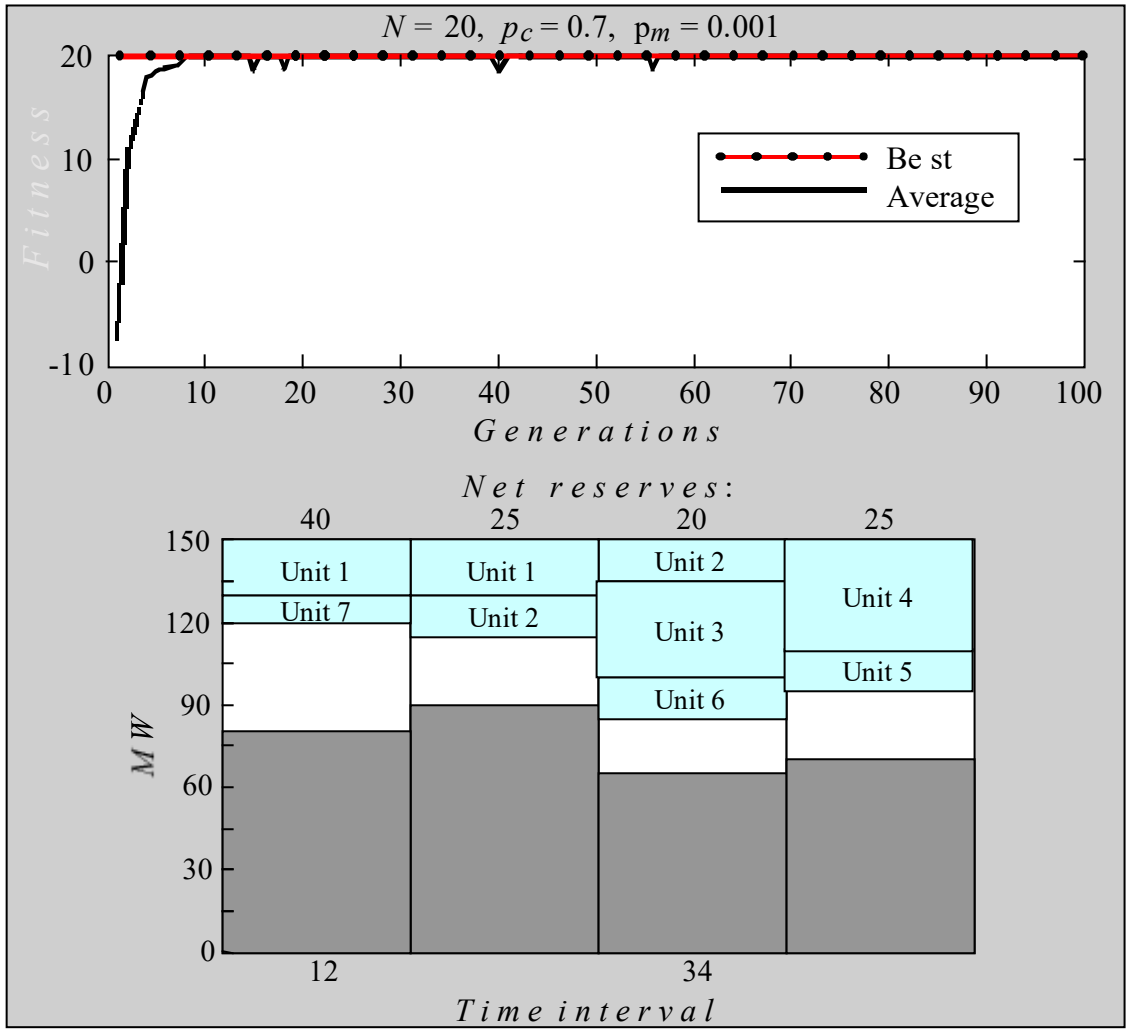
Case study: The mutation operator



Performance graphs and the best maintenance schedules created in a population of 20 chromosomes



Performance graphs and the best maintenance schedules created in a population of 20 chromosomes



Performance graphs and the best maintenance schedules created in a population of 100 chromosomes