Unit 6

Genetic Algorithm

Intelligence can be defined as the capability of a system to adapt its behaviour to an ever-changing environment. According to Alan Turing (Turing, 1950), the form or appearance of a system is irrelevant to its intelligence.

Course Content

- Introduction
- Genetic Algorithm
- Procedure of Genetic Algorithm
- The Working of Genetic Algorithm
- The logic Behind Genetic Algorithm
- Evolutionary Computing

Introduction

- **Algorithm**: An algorithm is a sequence of instructions to solve a problem. Most of the algorithms are static.

- **A Genetic Algorithm(GA)** is adaptive (dynamic) model of machine learning algorithm that derives its behavior from a metaphor of some of the mechanisms of evolution in nature.

Background

- On 1 July 1858, **Charles Darwin** , presented his **theory of evolution**. This day marks the beginning of a revolution in Biology.
- **Darwin**'s classical **theory of evolution** , together with Weismann's **theory of natural selection** and Mandel's concept of **genetics** , now represent the **neo-Darwinism**
- **Neo-Darwinism** is based on process of reproduction, mutation, competition and selection.

Background

- Evolution can be seen as a process leading to the maintenance of a population’s ability to *survive and reproduce* in a specific environment. This ability is called **evolutionary fitness**.
- Evolutionary fitness can also be viewed as a measure of the organism’s ability to anticipate changes in its environment.
- The fitness, or the quantitative measure of the ability to predict environmental changes and respond adequately, can be considered as the quality that is optimized in natural life

Evolutionary Computation

- Evolutionary Computation stimulates evolution on a computer. The result of such simulations is a sense of optimisation algorithms
- Optimisation iteratively improves the quality of solutions until an optimal, or near-optimal, solution is found
- The evolutionary approach is based on computational models of natural selection and genetics. We call them **evolutionary computation** ,an umbrella term that combines **genetic algorithms, evolution strategies** and **genetic programming**

Simulation of Natural Evolution

- All methods of evolutionary computation simulate natural evolution by creating a population of individuals, evaluating their fitness, generating a new population through genetic operations, and repeating this process a number of times.
- We focus on **Genetic Algorithm** as most of the other algorithms can be viewed as variations of genetic algorithms.

Genetic Algorithm

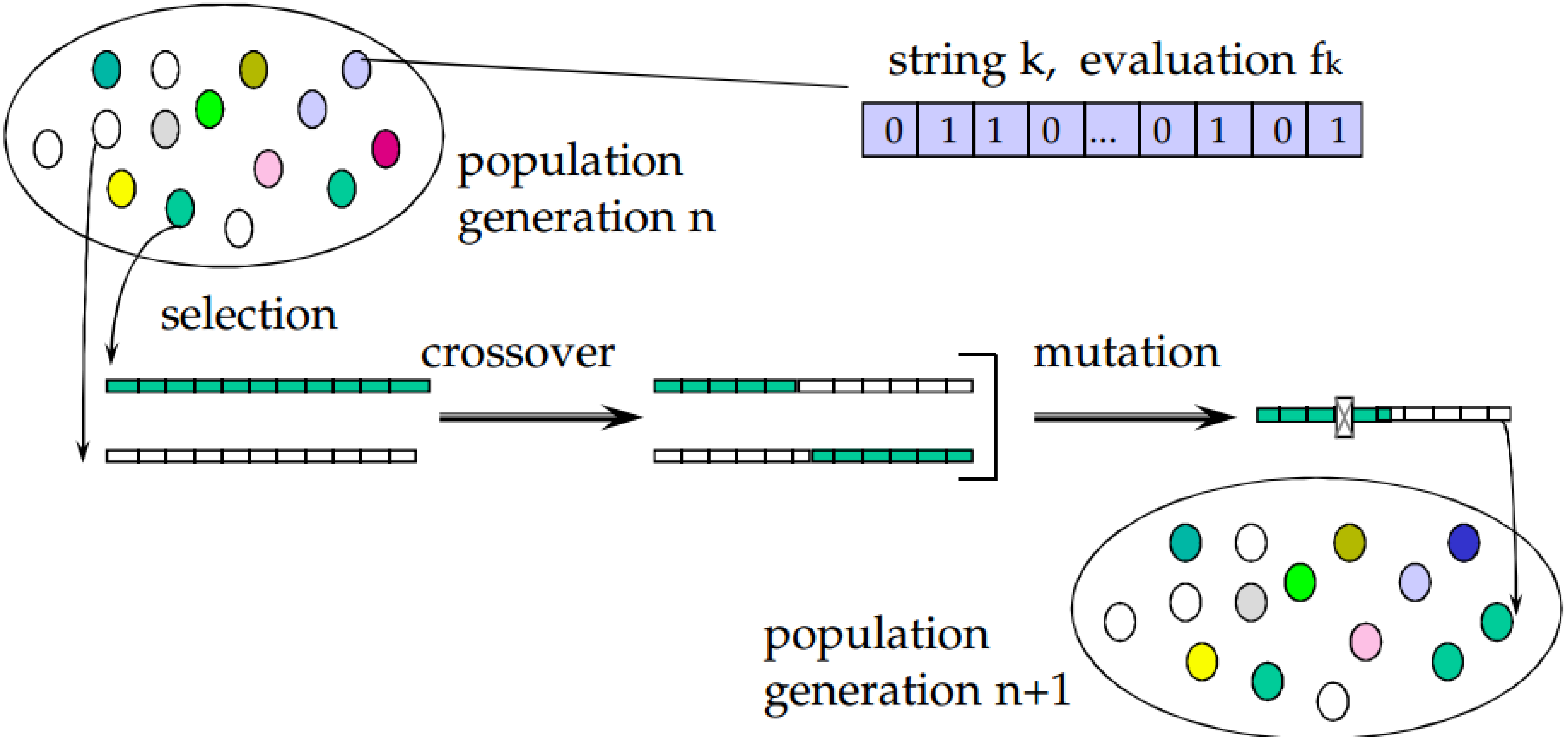
- In early 1970s John Holland introduced the concept of genetic algorithm
- His aim was to make computers do what nature does. Holland was concerned with algorithms that manipulate strings of binary digits
- Each artificial “chromosomes” consists of a number of “genes”, and each gene is represented by 0 or 1

1	0	1	1	0	1	0	0	0	0	0	1	0	1	0	1

Genetic Algorithm

- Two mechanisms link a GA to the problem it is solving : **Encoding** and **Evaluation**
- The GA uses a measure of fitness of individual chromosomes to carry out reproduction. As reproduction takes place, the crossover operator exchanges part of two single chromosomes and the mutation operator changes the gene value in some randomly chosen location of the chromosome

Basic Genetic Algorithm



Genetic Algorithm

GA Operators and Parameters

- **Fitness function:** The fitness function is defined over the genetic representation and measures the *quality* of the represented solution.
- **Selection Operator:** Selects parents for reproduction based on relative fitness of candidates in the population
  - Roulette Wheel Selection
  - Ranking Selection

Genetic Algorithm

- **Crossover Operator:**
  - Exchanges part of chromosome between two parent chromosomes with some crossover rate(probability), typically 0.4 – 0.8
  - The main operator to provide exploitation in search building up good genes in chromosome
    - **One Point Crossover:** 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 is created
    - **Two Point Crossover:** randomly chooses two crossover points in two parent chromosomes, and then exchanges the chromosome parts between these points. As a result, two new offspring are created.

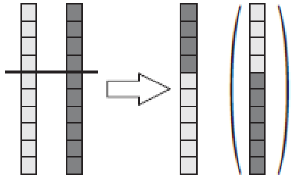


Fig. .a: Single-point Crossover (SPX).

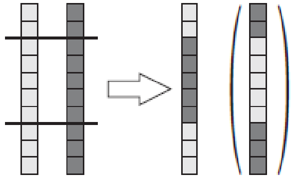


Fig. .b: Two-point Crossover (TPX).

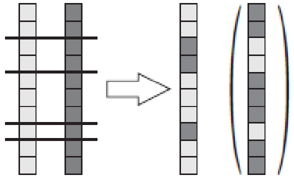
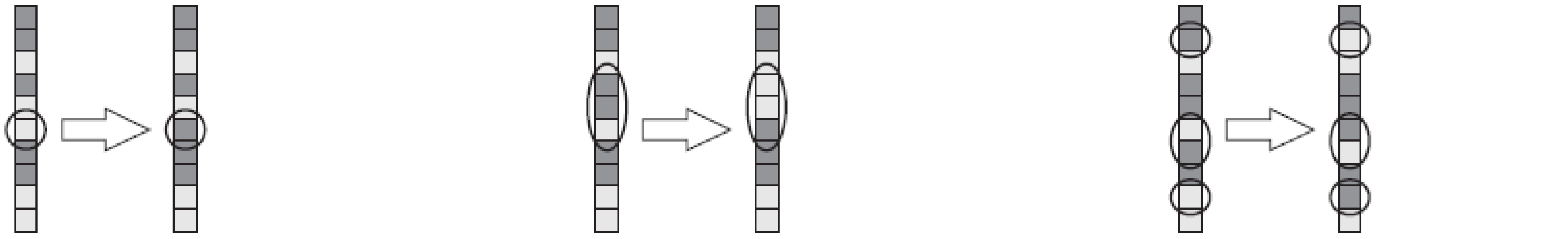


Fig. .c: Multi-point Crossover (MPX).

Genetic Algorithm

- **Mutation Operator:**
  - Changes a randomly selected gene in the chromosome
  - mimics random changes in genetic code
  - Background operator to provide exploration in search to avoid being trapped on a local optimum
  - Mutation probability is quiet small in nature and is kept low for GAs, typically in the range between [0.001 – 0.01] or by formula :

    P(m) = 1/no, of bits in Chromosomes



a: Single-gene mutation. b: Multi-gene mutation c: Multi-gene mutation

Genetic Algorithm

- **Elitism Approach :** Saves the best individual in next generation
- **Basic GA Parameters:**
  - Population size
  - Crossover rate (Probability)
  - Mutation Rate (Probability)
  - Number of Generation (a Stopping Criterion)

Steps in Genetic Algorithm

1. Represent the problem variable as a chromosome of a fixed length, choose the size of a chromosome population *N*, the crossover probability *pc* and the mutation probability *pm*.
2. Define a fitness function to measure the fitness of an individual chromosome in the problem domain.
3. Randomly generate an initial population of chromosomes of size *N*: *x*1, *x*2,.., *xN*
4. Calculate the fitness of each individual chromosome: *f*(*x*1), *f*(*x*2), . . . , *f*(*xN*)
5. Select a pair of chromosomes for mating from the current population based on their fitness.
6. Create a pair of offspring chromosomes by applying the genetic operators − **crossover** and **mutation**.
7. Place the created offspring chromosomes in the new population.
8. Repeat *Step 5* until the size of the new chromosome population becomes equal to the size of the initial population, *N*.
9. Replace the initial (parent) chromosome population with the new (offspring) population.
10. Go to *Step 4*, and repeat the process until the termination criterion is satisfied

Genetic Algorithm : Case Study

- A simple Example will help understand how a GA works. Let is 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
- Mathematically this is an optimisation problem : find the value of variable x so that max($15x - x^2$) such that $0 \leq x \leq 15 \; x \forall \; integers$

Representation (Encoding)

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

GA : Case Study

- **Fitness Function** : The Fitness function in put example is defined by
$$\boldsymbol{f(x) = 15x - x^2}$$
  the solution (i.e. value of x is better when this value if high
- **GA Operator and Parameter:** suppose the size of the chromosome population is *N* is 6, the crossover probability $p_c = 0.7$, and mutation probability $p_m = 0.001$

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

Fig: initially randomly generated population of Chromosomes

GA: Case Study

- 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

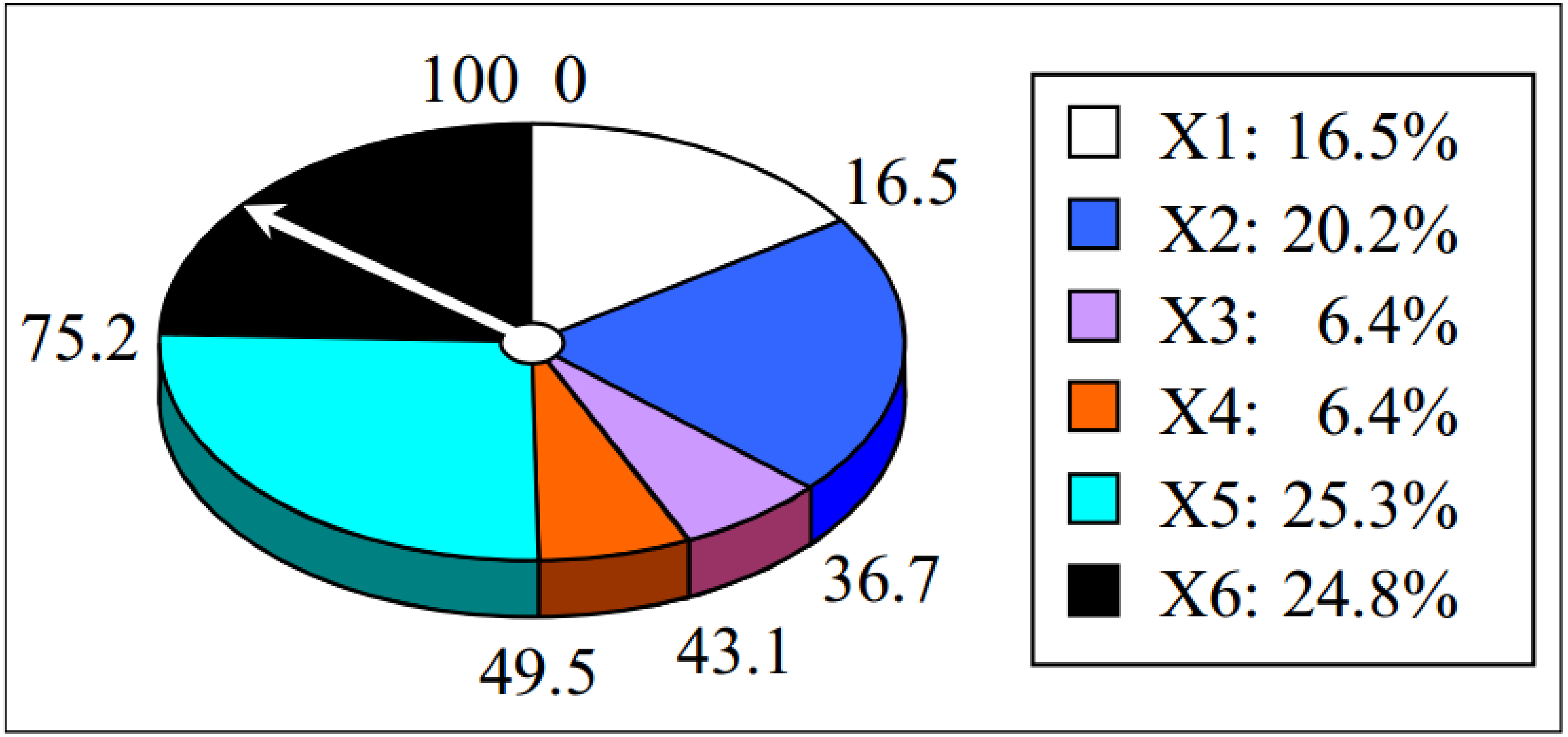
GA: Case Study

- 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
- If needed **Mutation** is also carried out to avoid being trapped in local minimum

GA: Case Study

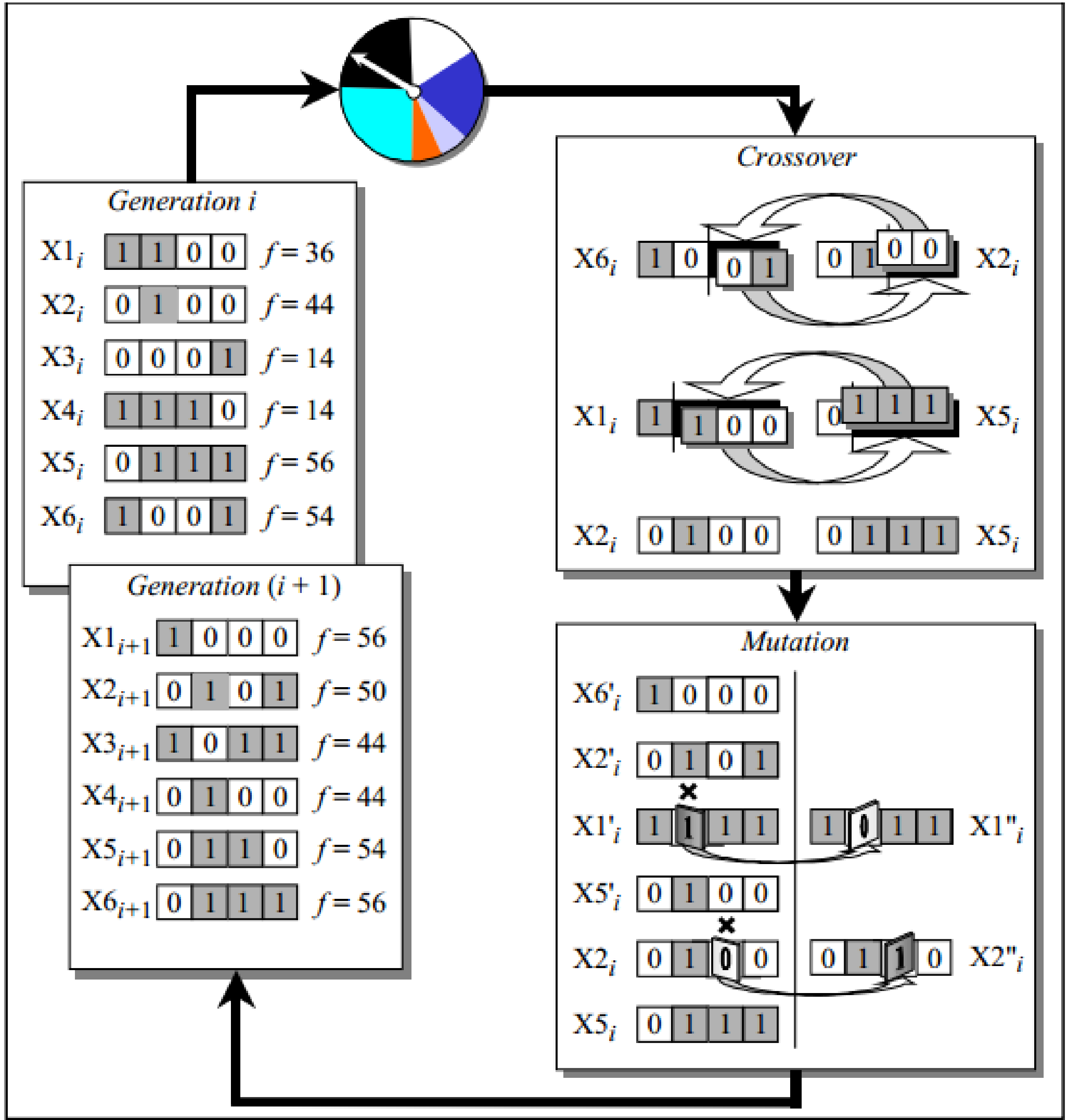
Roulette-Wheel Selection

The most commonly used chromosome selection technique is the roulette wheel selection.

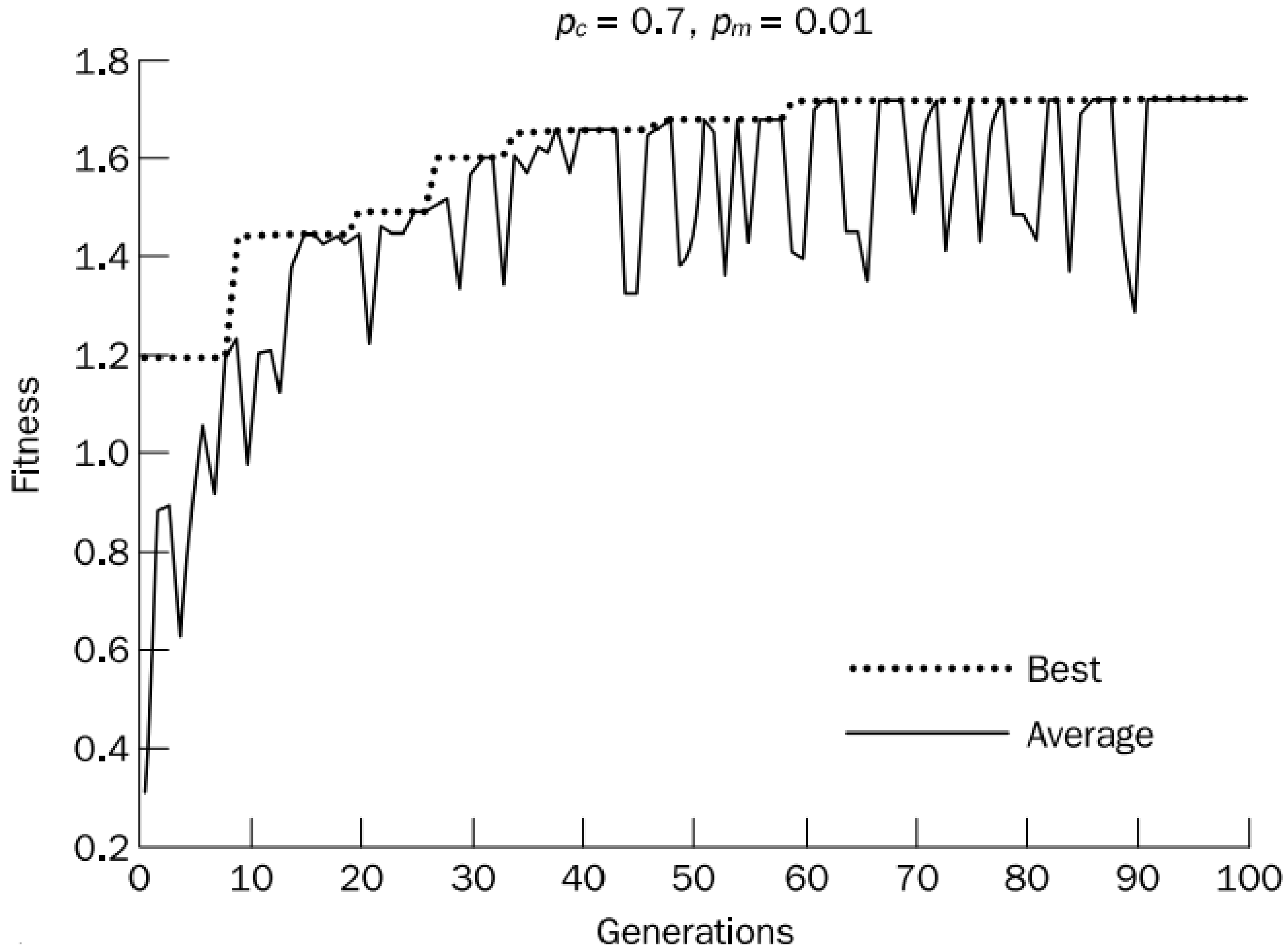


GA: Case Study

The Genetic Algorithm Cycle



Genetic Algorithm



GA : Case Study 2

Example of Selection

Evolutionary Algorithms is to maximize the function $f(x) = x^2$ with x in the integer interval $[0, 31]$, i.e., $x = 0, 1, \ldots 30, 31$.

1. The first step is encoding of chromosomes; use binary representation for integers; 5-bits are used to represent integers up to 31.
2. Assume that the population size is 4.
3. Generate initial population at random. They are chromosomes or genotypes; e.g., 01101, 11000, 01000, 10011.
4. Calculate fitness value for each individual.
   (a) Decode the individual into an integer (called phenotypes),
   01101 → 13; 11000 → 24; 01000 → 8; 10011 → 19;
   (b) Evaluate the fitness according to $f(x) = x^2$,
   13 → 169; 24 → 576; 8 → 64; 19 → 361.
5. Select parents (two individuals) for crossover based on their fitness in p_i. Out of many methods for selecting the best chromosomes, if **roulette-wheel** selection is used, then the probability of the i^{th} string in the population is $p_i = F_i \;/\; (\sum_{j=1}^{n} F_j)$, where

   F_i is fitness for the string i in the population, expressed as $f(x)$
   p_i is probability of the string i being selected,
   n is no of individuals in the population, is population size, $n=4$
   $n * p_i$ is expected count

String No	Initial Population	X value	Fitness Fi $f(x) = x^2$	p i	Expected count N * Prob *i*
1	0 1 1 0 1	13	169	0.14	0.58
2	1 1 0 0 0	24	576	0.49	1.97
3	0 1 0 0 0	8	64	0.06	0.22
4	1 0 0 1 1	19	361	0.31	1.23
Sum			1170	1.00	4.00
Average			293	0.25	1.00
Max			576	0.49	1.97

GA : Case Study 2

The string no 2 has maximum chance of selection.

6. Produce a new generation of solutions by picking from the existing pool of solutions with a preference for solutions which are better suited than others:

We divide the range into four bins, sized according to the relative fitness of the solutions which they represent.

Strings	*Prob i*	*Associated Bin*
0 1 1 0 1	0.14	0.0 . . . 0.14
1 1 0 0 0	0.49	0.14 . . . 0.63
0 1 0 0 0	0.06	0.63 . . . 0.69
1 0 0 1 1	0.31	0.69 . . . 1.00

By generating 4 uniform (0, 1) random values and seeing which bin they fall into we pick the four strings that will form the basis for the next generation.

Random No	*Falls into bin*	*Chosen string*
0.08	0.0 . . . 0.14	0 1 1 0 1
0.24	0.14 . . . 0.63	1 1 0 0 0
0.52	0.14 . . . 0.63	1 1 0 0 0
0.87	0.69 . . . 1.00	1 0 0 1 1

GA : Case Study 2

7. Randomly pair the members of the new generation

   Random number generator decides for us to mate the first two strings together and the second two strings together.

8. Within each pair swap parts of the members solutions to create offspring which are a mixture of the parents :

   For the first pair of strings: **0 1 1 0 1 , 1 1 0 0 0**

   – We randomly select the crossover point to be after the fourth digit. Crossing these two strings at that point yields:

   0 1 1 0 1 ⇒ 0 1 1 0 |1 ⇒ 0 1 1 0 0

   1 1 0 0 0 ⇒ 1 1 0 0 |0 ⇒ 1 1 0 0 1

   For the second pair of strings: **1 1 0 0 0 , 1 0 0 1 1**

   – We randomly select the crossover point to be after the second digit. Crossing these two strings at that point yields:

   1 1 0 0 0 ⇒ 1 1 |0 0 0 ⇒ 1 1 0 1 1

   1 0 0 1 1 ⇒ 1 0 |0 1 1 ⇒ 1 0 0 0 0

GA : Case Study 2

9. Randomly mutate a very small fraction of genes in the population :

With a typical mutation probability of per bit it happens that none of the bits in our population are mutated.

10. Go back and re-evaluate fitness of the population (new generation) :

This would be the first step in generating a new generation of solutions. However it is also useful in showing the way that a single iteration of the genetic algorithm has improved this sample.

String No	Initial Population (chromosome)	X value (Pheno types)	Fitness $f(x) = x^2$	Prob i (fraction of total)	Expected count
1	0 1 1 0 0	12	144	0.082	0.328
2	1 1 0 0 1	25	625	0.356	1.424
3	1 1 0 1 1	27	729	0.415	1.660
4	1 0 0 0 0	16	256	0.145	0.580
Total (sum)			1754	1.000	4.000
Average			439	0.250	1.000
Max			729	0.415	1.660

Observe that :

1. Initial populations : At start step 5 were

   0 1 1 0 1 , 1 1 0 0 0 , 0 1 0 0 0 , 1 0 0 1 1

   After one cycle, new populations, at step 10 to act as initial population

   0 1 1 0 0 , 1 1 0 0 1 , 1 1 0 1 1 , 1 0 0 0 0

2. The total fitness has gone from **1170** to **1754** in a single generation.
3. The algorithm has already come up with the string 11011 (i.e **x = 27**) as a possible solution.

Genetic Algorithm

Genetic algorithms are used to solve many large problems including:

-Scheduling

- Transportation

-Chemistry, Chemical Engineering

- Layout and circuit design

-Medicine

-Data Mining and Data Analysis

-Economics and Finance

-Networking and Communication

- Game etc.

Genetic Algorithm

GA Advantages and Disadvantages

Advantages:

- It can solve every optimization problem which can be described with the chromosome encoding.
- It solves problems with multiple solutions.
- Since the genetic algorithm execution technique is not dependent on the error surface, we can solve multi-dimensional, non-differential, non-continuous, and even non-parametrical problems.
- Structural genetic algorithm gives us the possibility to solve the solution structure and solution parameter problems at the same time by means of genetic algorithm.
- Genetic algorithm is a method which is very easy to understand and it practically does not demand the knowledge of mathematics.
- Genetic algorithms are easily transferred to existing simulations and model

Disadvantages

- May be Slow
- May be drop of the quality because of crossover.

References:

- Artificial Intelligence A Guide to Intelligent Systems: Michael Negnevitsky [2ed]

```
# Initialization
population = generate_random_population()
for generation in range(num_generations):
    # Evaluate fitness
    fitness_scores = evaluate_fitness(population)
    # Selection
    parents = select_parents(population, fitness_scores)
    # Crossover
    offspring = crossover(parents)
    # Mutation
    mutate(offspring)
    # Replacement
    population = replace(population, offspring)

# Final result
best_route = get_best_route(population)
```