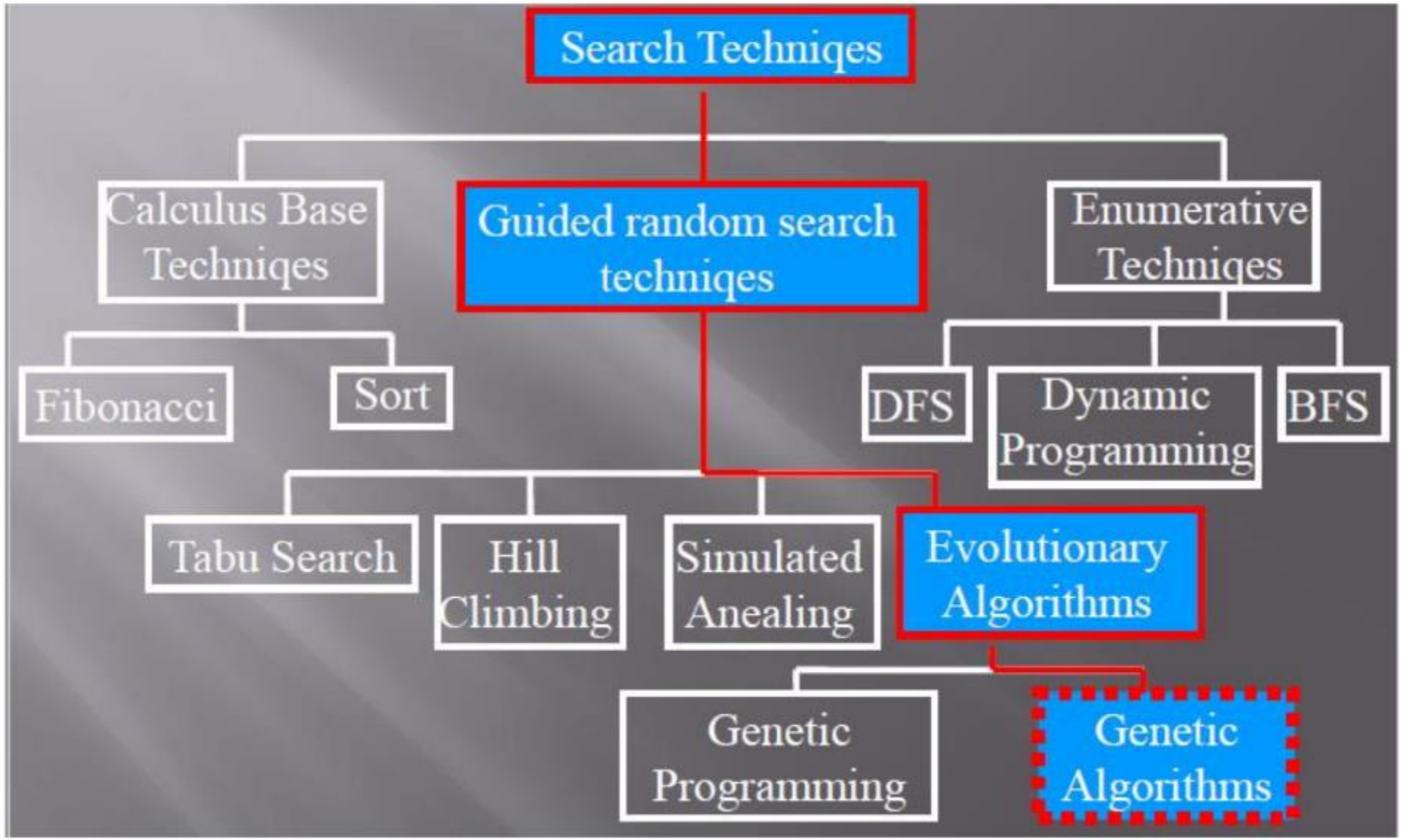
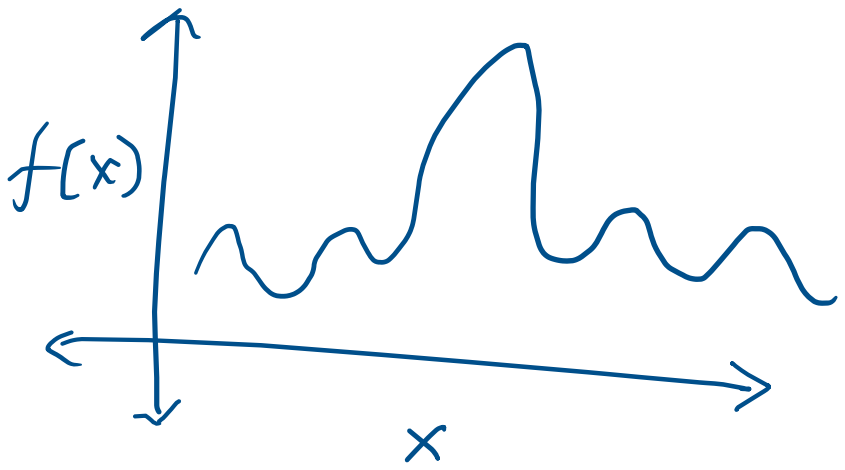
Genetic Algorithms

Classes of Search Techniques



→ To solve optimization Problem.

$f(x)$.



Now here, for some value of x, $f(x)$ will contain global optima and for some it will contain local optima values.

$\Rightarrow$ We want to find for which value of x $f(x)$ minimizes $\rightarrow$ we have to search the search space to obtain appropriate value of x.

→ Mathematically

$f(x_1 \ldots x_n)$ → function

where f is a objective function.

and $x_1 \ldots x_n$ → parameters.

→ minimize $f(x_1 \ldots x_n)$ subject to certain constraints.

⇒ This problem cannot be solved in normal time.

→ Traditional optimization techniques cannot solve such problems in real time.

For example, in a population of 5 crore people, there are only 20 knowledgable persons. Finding those 20 people is not possible in real time.

=> So, there Genetic Algorithms come into picture, where

Limitations of the traditional optimization approaches

- Limitations:
  - Computationally expensive.
  - For a discontinuous objective function, methods may fail.
  - Method may not be suitable for parallel computing.
  - Discrete (integer) variables are difficult to handle.
  - Methods may not necessarily adaptive.
- Evolutionary algorithms have been evolved to address the above mentioned limitations of solving optimization problems with traditional approaches.

Evolutionary Algorithms

The algorithms, which follow some biological and physical behaviors:

Biologic behaviors:

- Genetics and Evolution –> Genetic Algorithms (GA)
- Behavior of ant colony –> Ant Colony Optimization (ACO)
- Human nervous system –> Artificial Neural Network (ANN)

In addition to that there are some algorithms inspired by some physical behaviors:

Physical behaviors:

- Annealing process –> Simulated Annealing (SA)
- Swarming of particle –> Particle Swarming Optimization (PSO)
- Learning –> Fuzzy Logic (FL)

Genetic Algorithm

It is a subset of evolutionary algorithm:

- Ant Colony optimization
- Swarm Particle Optimization

Models biological processes:

- Genetics
- Evolution

To optimize highly complex objective functions:

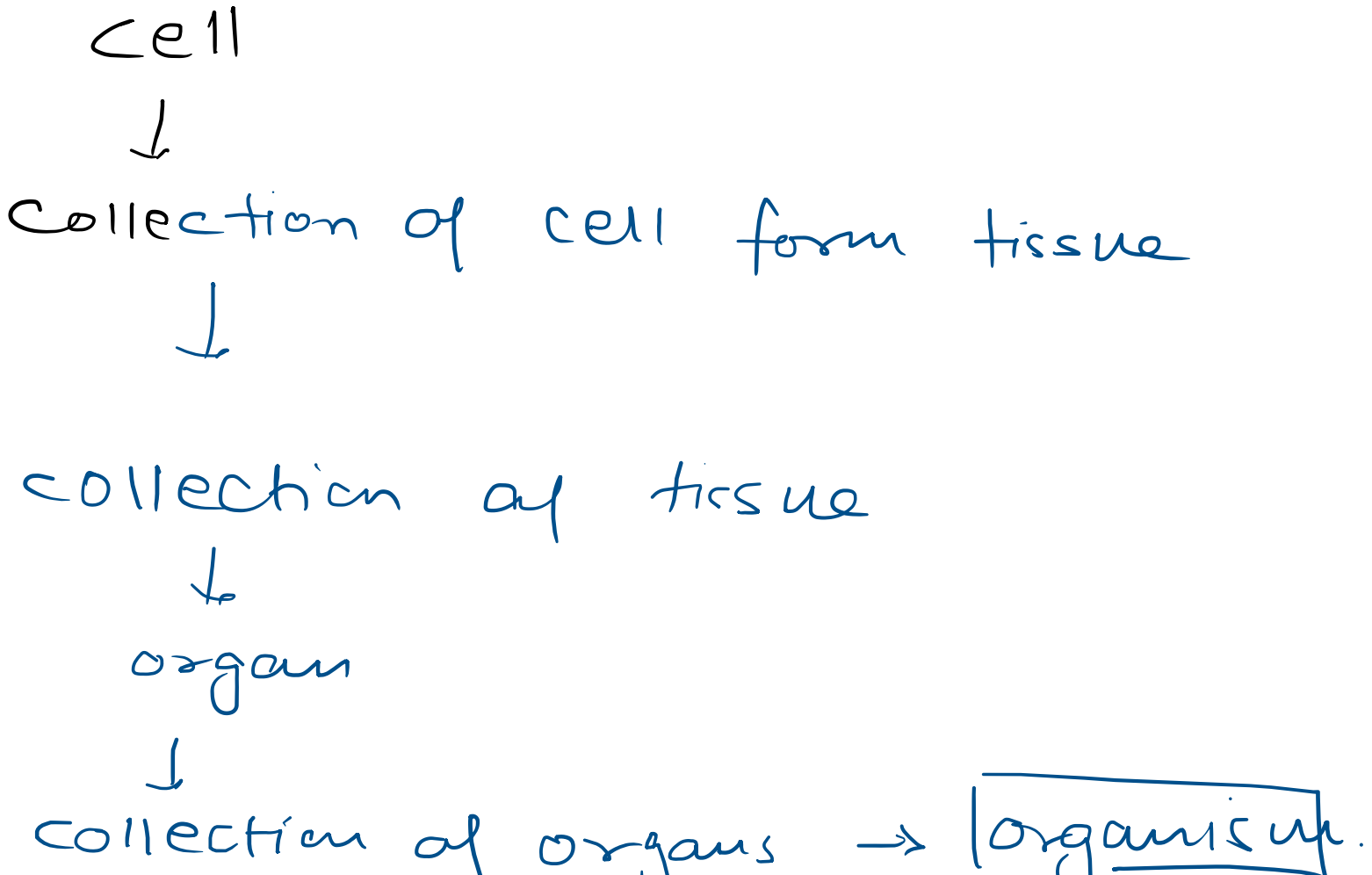
- Very difficult to model mathematically
- **NP-Hard** (also called combinatorial optimization) problems (which are computationally very expensive)
- Involves large number of parameters (discrete and/or continuous)

Firs time itriduced by Ptrof. John Holland (of Michigan University, USA, 1965).
But, the first article on GA was published in 1975.

Principles of GA based on two fundamental biological processes:

- **Genetics:** Gregor Johan Mendel (1865)
- **Evolution:** Charles Darwin (1875)

A brief idea of Genetics



Collection of organism

↓

Population

→

Collection of species of similar characteristics (Humans)

only → Lion

only Tiger.

Particular organism → identified by chromosome pair

- For a particular specie, number of chromosomes is fixed.

  Examples
  - Mosquito: 6
  - Frogs: 26
  - Human: 46
  - Goldfish: 94

    etc.

chromosome for human = 23 pairs

↓

Collection of these chromosome

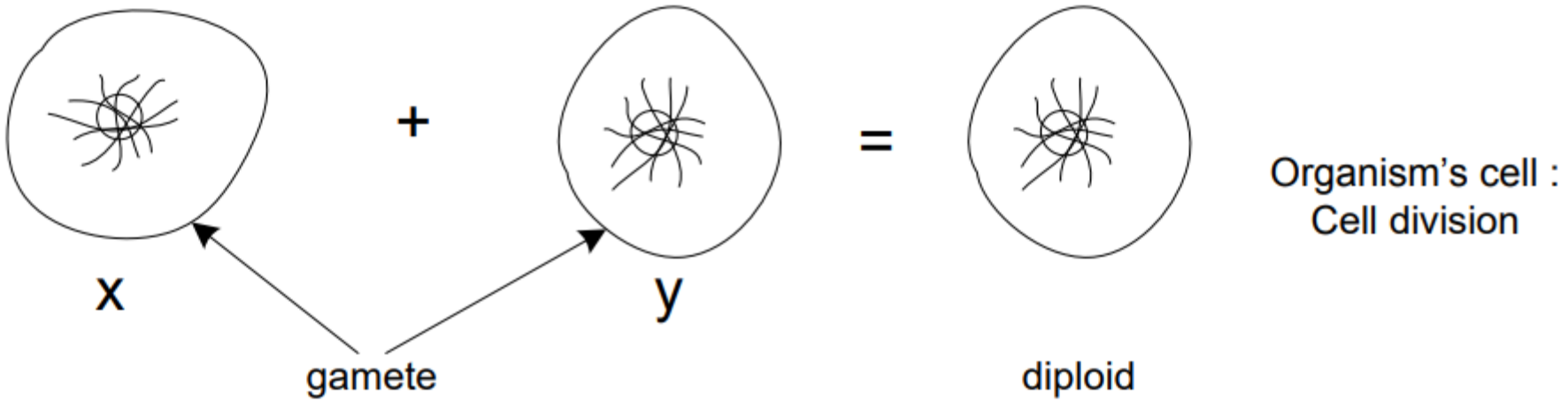
↓

population

Every chromosome is built up of million of genes.

How Humans replicate

Reproduction



Organism's cell :
Cell division

haploid	(Reproductive cell has half the number of chromosomes)
diploid	Each chromosome from both haploids are combined to have full numbers

Offspring produced by crossover.

survival of that offspring

↓

Evaluation

survival of the fittest

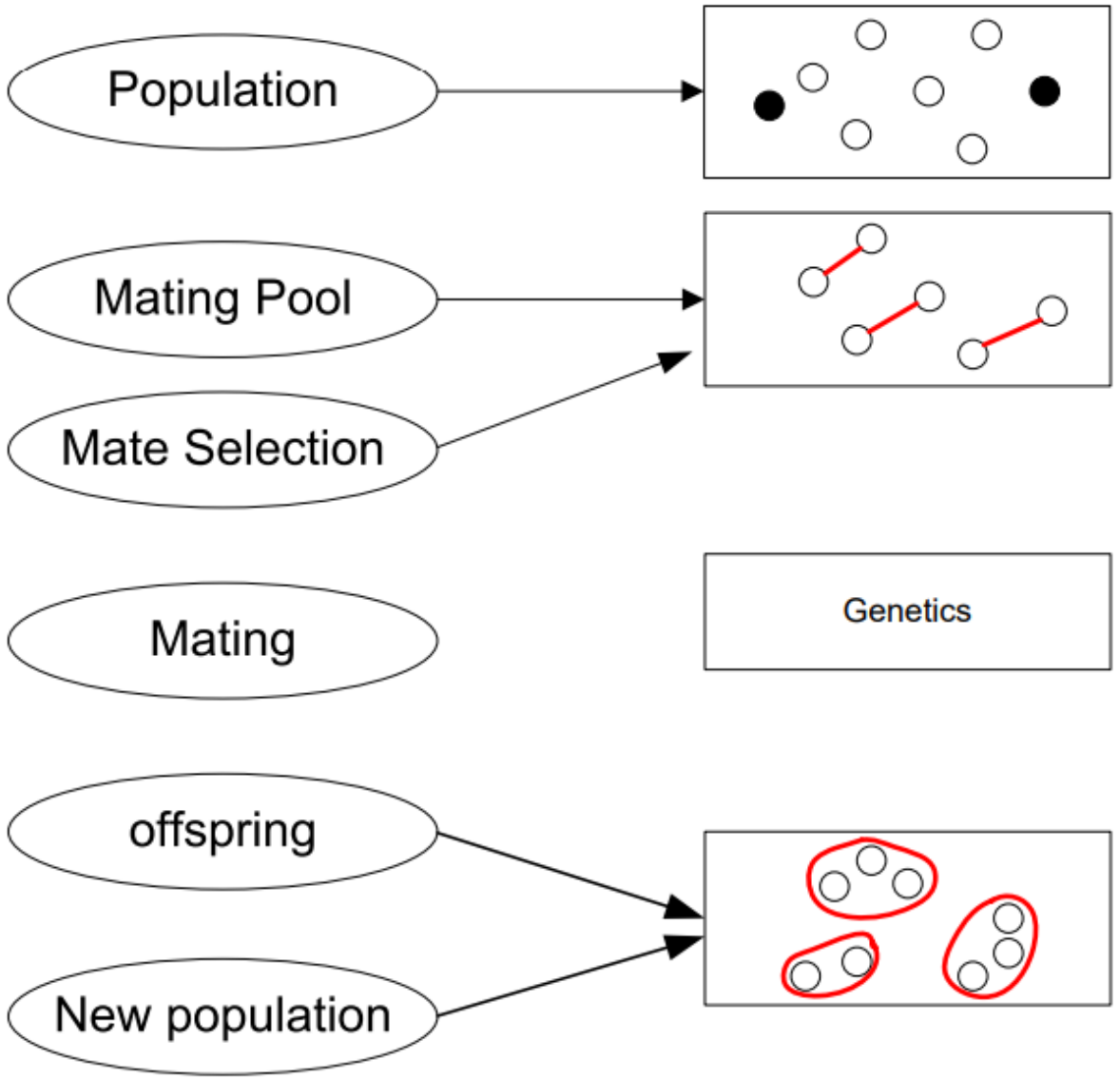
→ Individuals that adapt better to the given environment have higher chance of survival while the others die over time.

Ex. Dinosaur

So, selection of the fittest is done to create the next population & it goes on.

⇒ Darwinian Evolution

Biological process : A quick overview

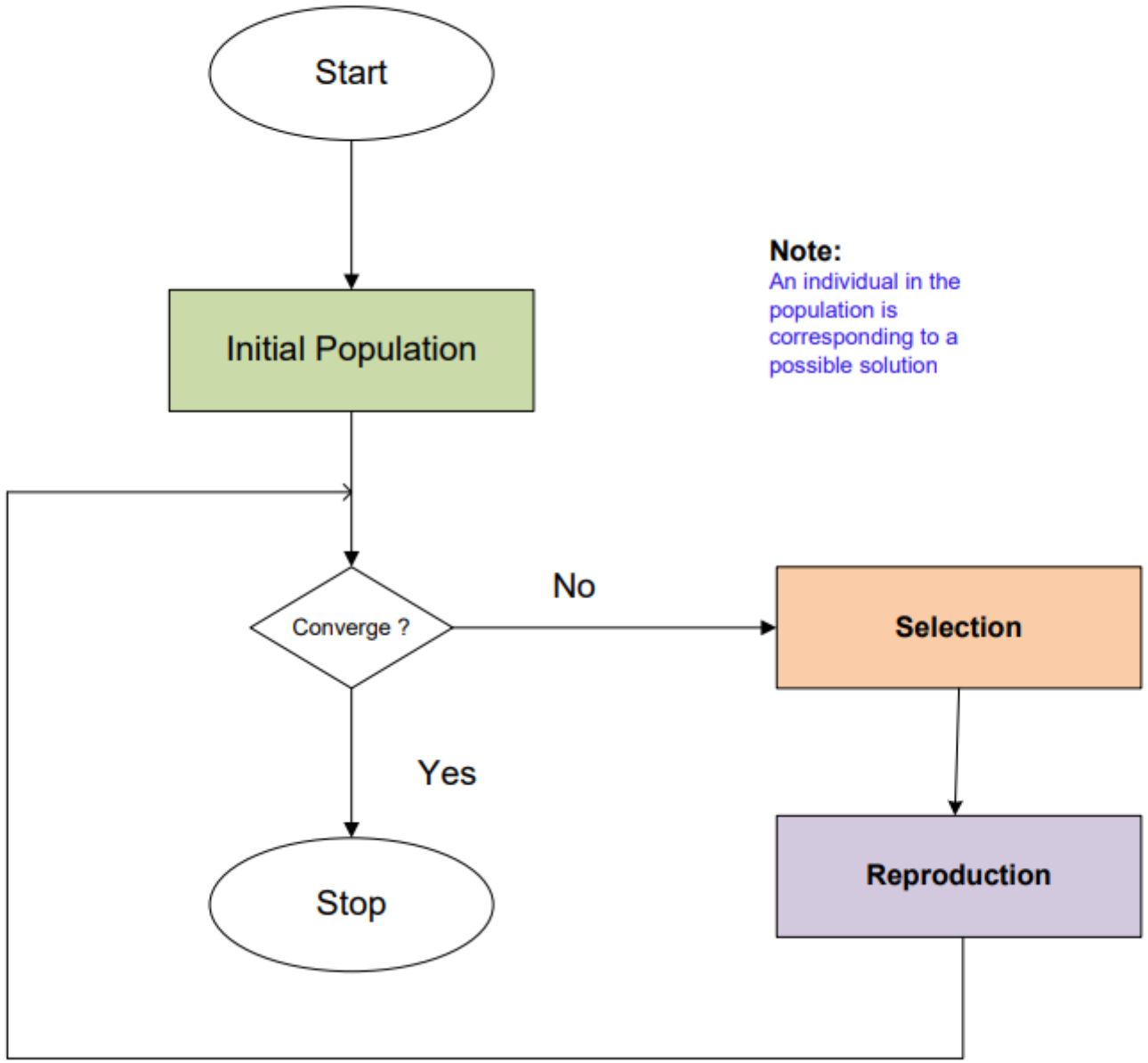


GA

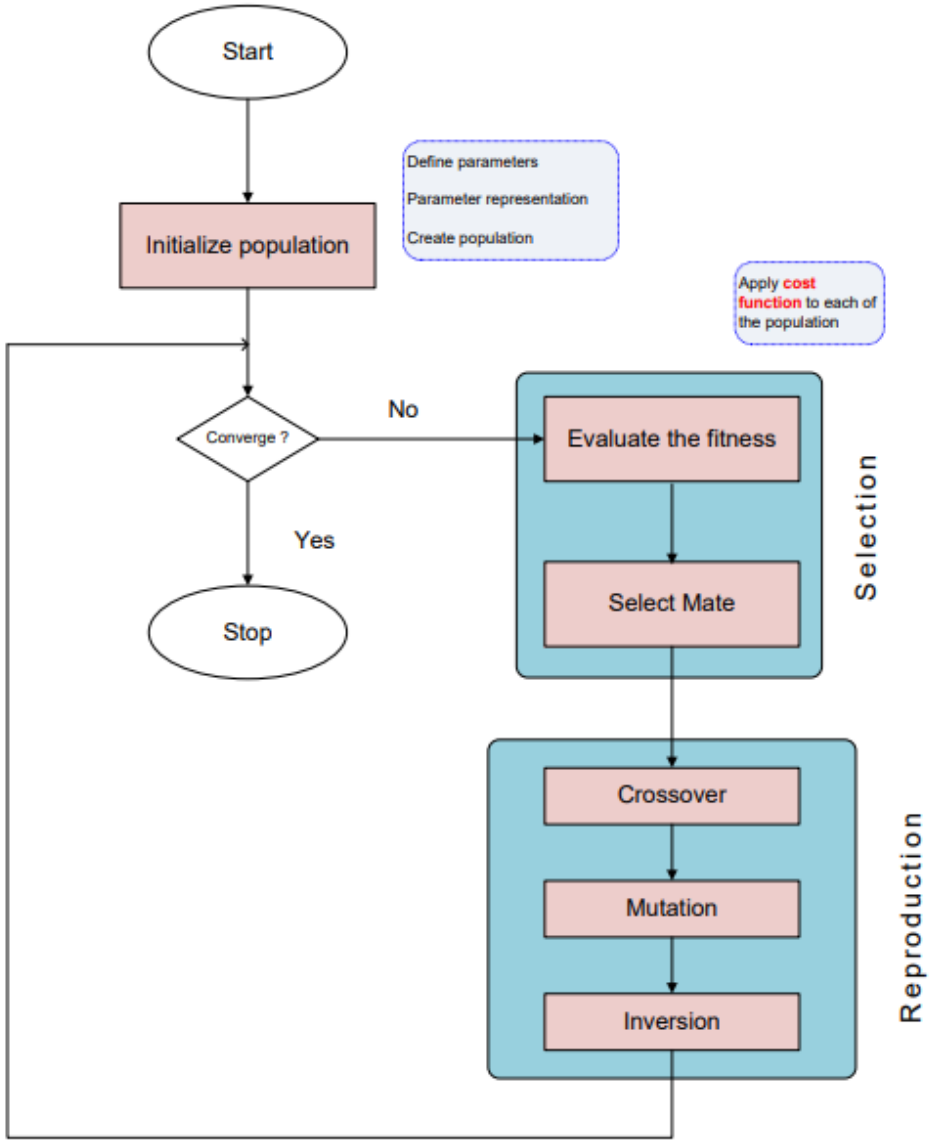
↳ Genetics
↳ Evolution } Both of this to solve optimization Problems.

- Genetic algorithm is a population-based probabilistic search and optimization techniques, which works based on the mechanisms of natural genetics and natural evaluation.

Framework of GA



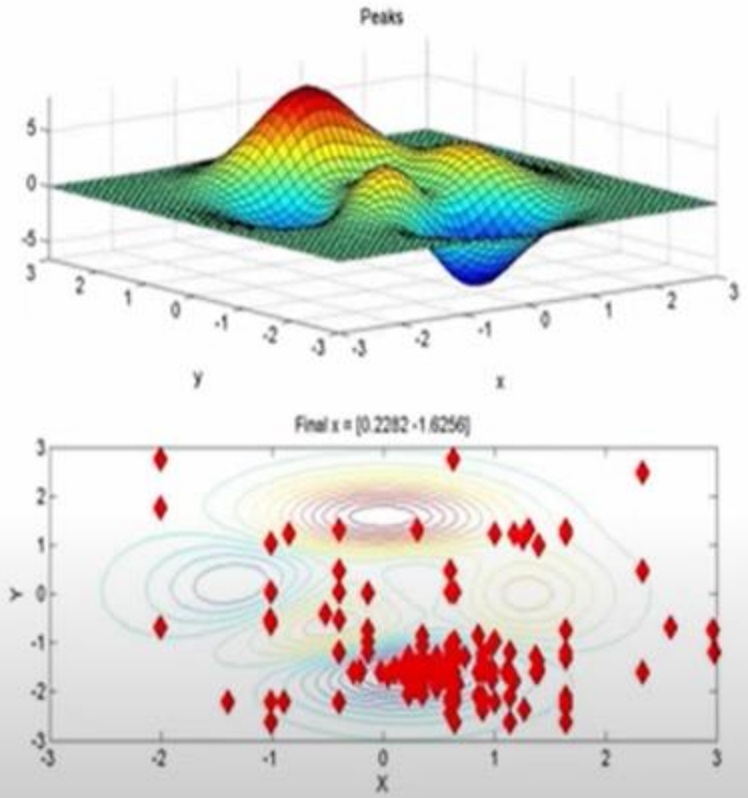
Framework of GA: A detail view



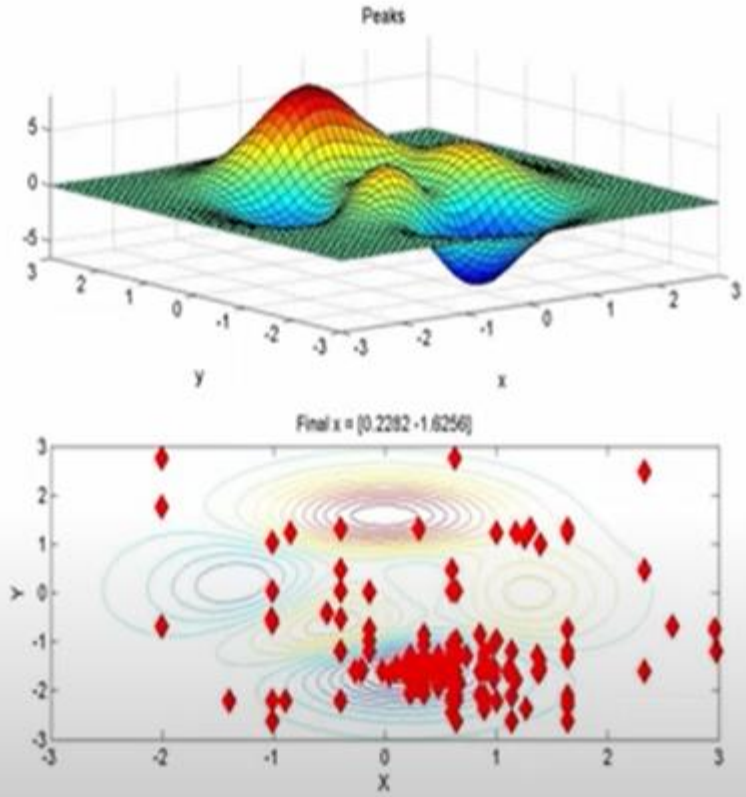
An Example :

What is a Genetic Algorithm?

- Uses concepts from *evolutionary biology*



- Uses concepts from *evolutionary biology*
- Start with an initial generation of candidate solutions that are tested against the objective function
- Subsequent generations evolve from the 1st through *selection*, *crossover* and *mutation*



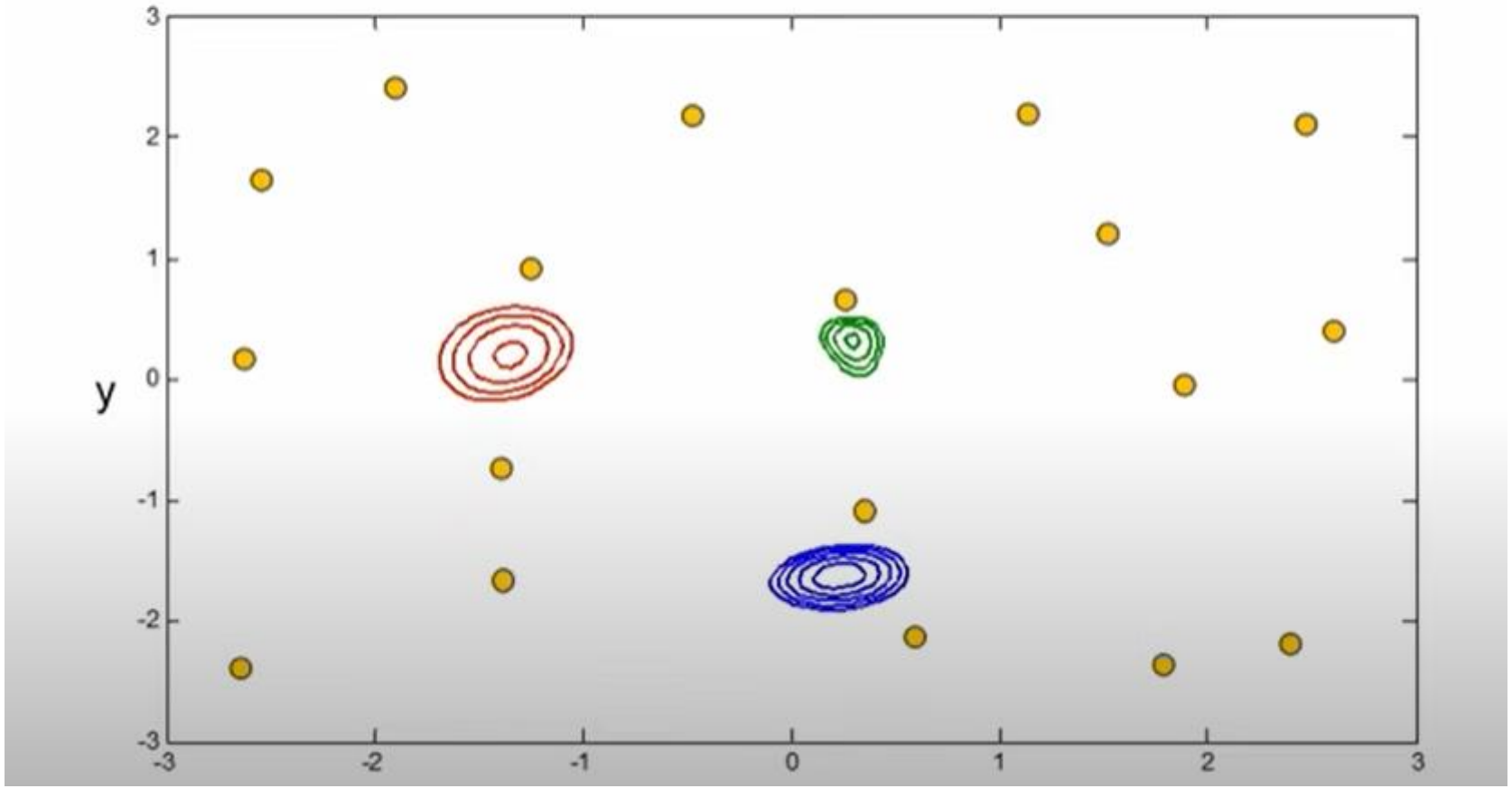
- Selection
  - *Retain* the best performing bit strings from one generation to the next. *Favor these for reproduction*
  - parent1 = [1 0 1 0 0 1 1 0 0 0]
  - parent2 = [1 0 0 1 0 0 1 0 1 0]

- Crossover
  - parent1 = [1 0 1 0 0 1 1 0 0 0]
  - parent2 = [1 0 0 1 0 0 1 0 1 0]
  - child = [1 0 0 0 0 1 1 0 1 0]

- Mutation
  - parent = [1 0 1 0 0 1 1 0 0 0]
  - child = [0 1 0 1 0 1 0 0 0 1]

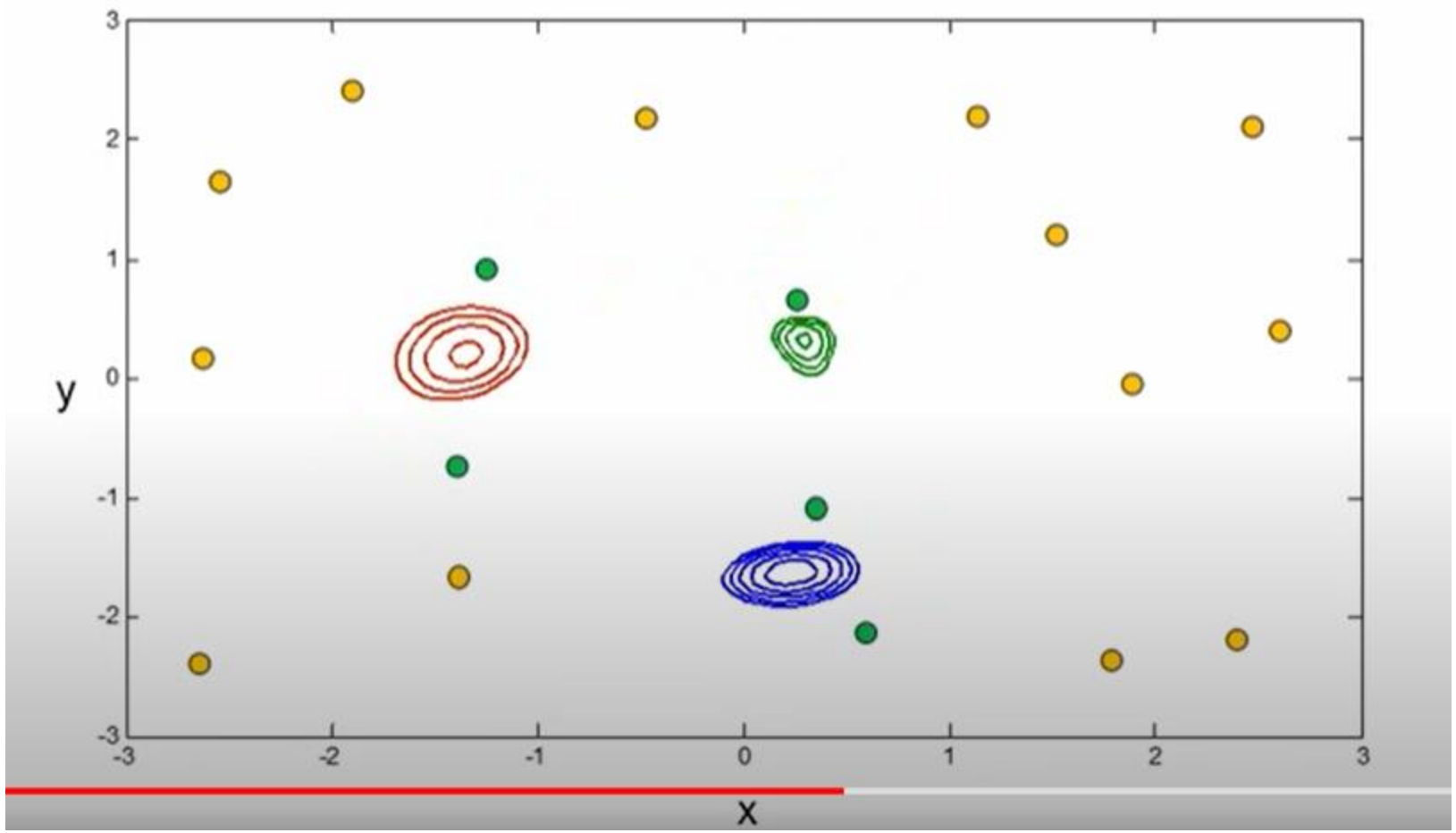
Genetic Algorithm – Iteration 1

Evaluate initial population



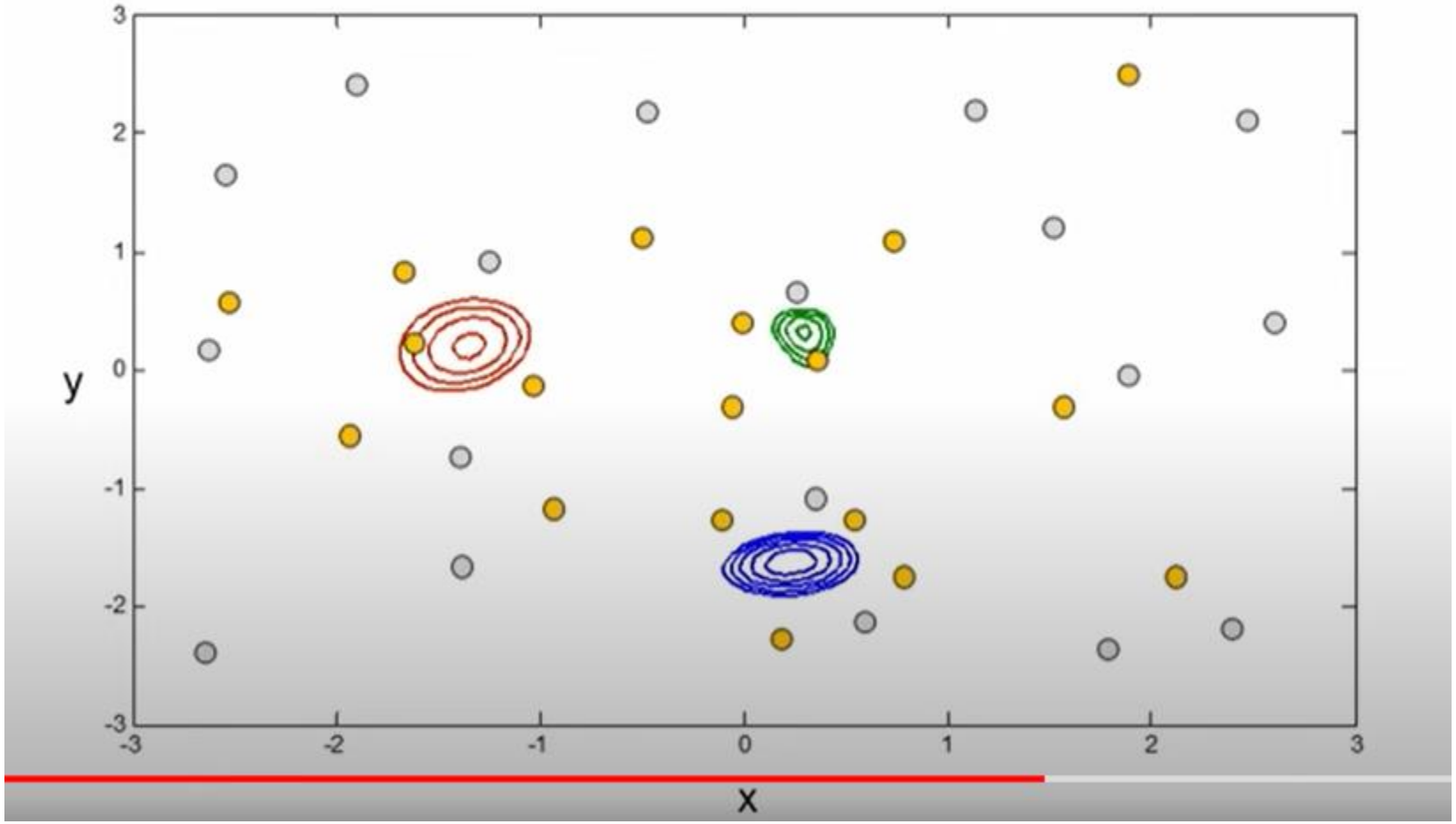
Genetic Algorithm – Iteration 1

Select a few good solutions for reproduction

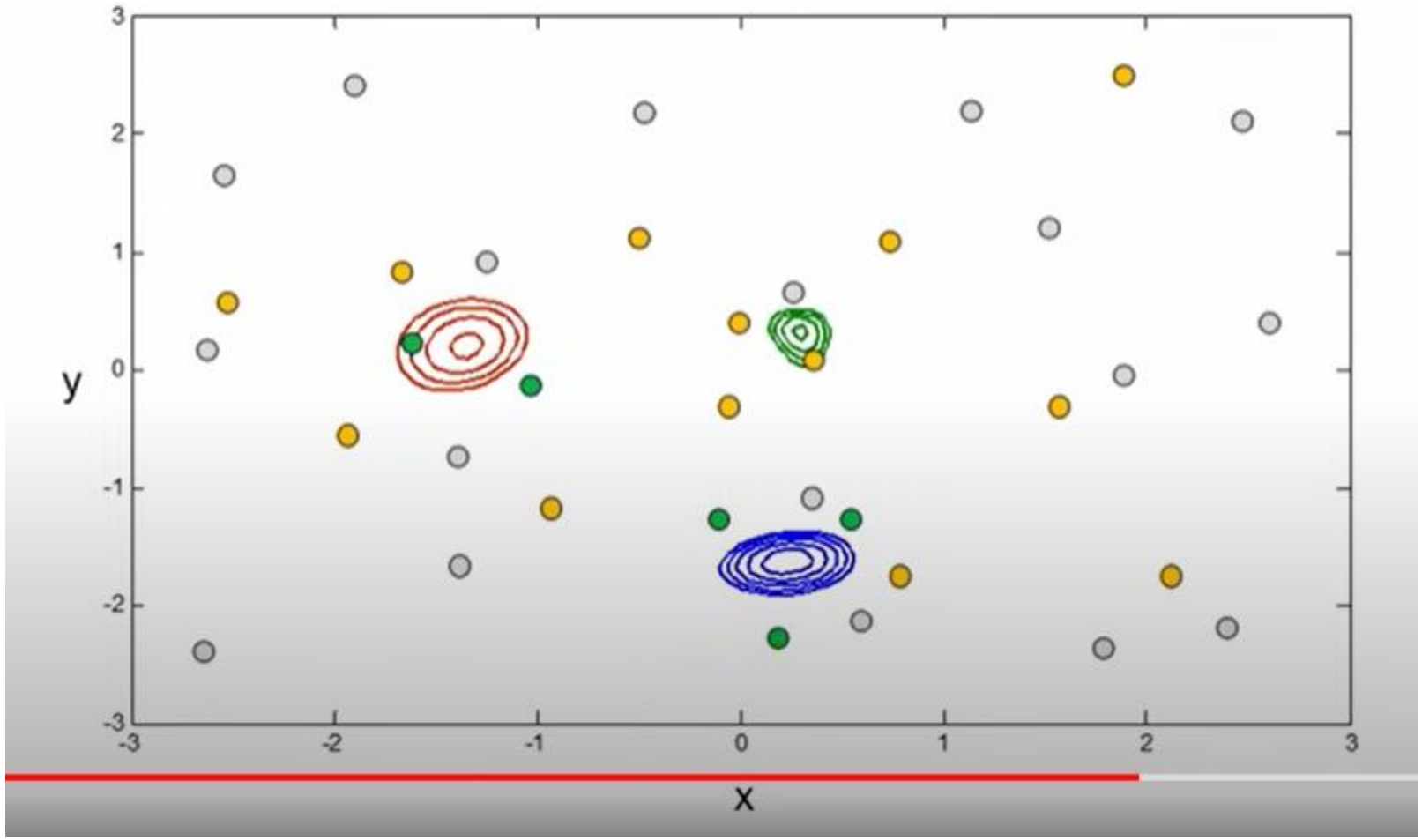


Genetic Algorithm – Iteration 2

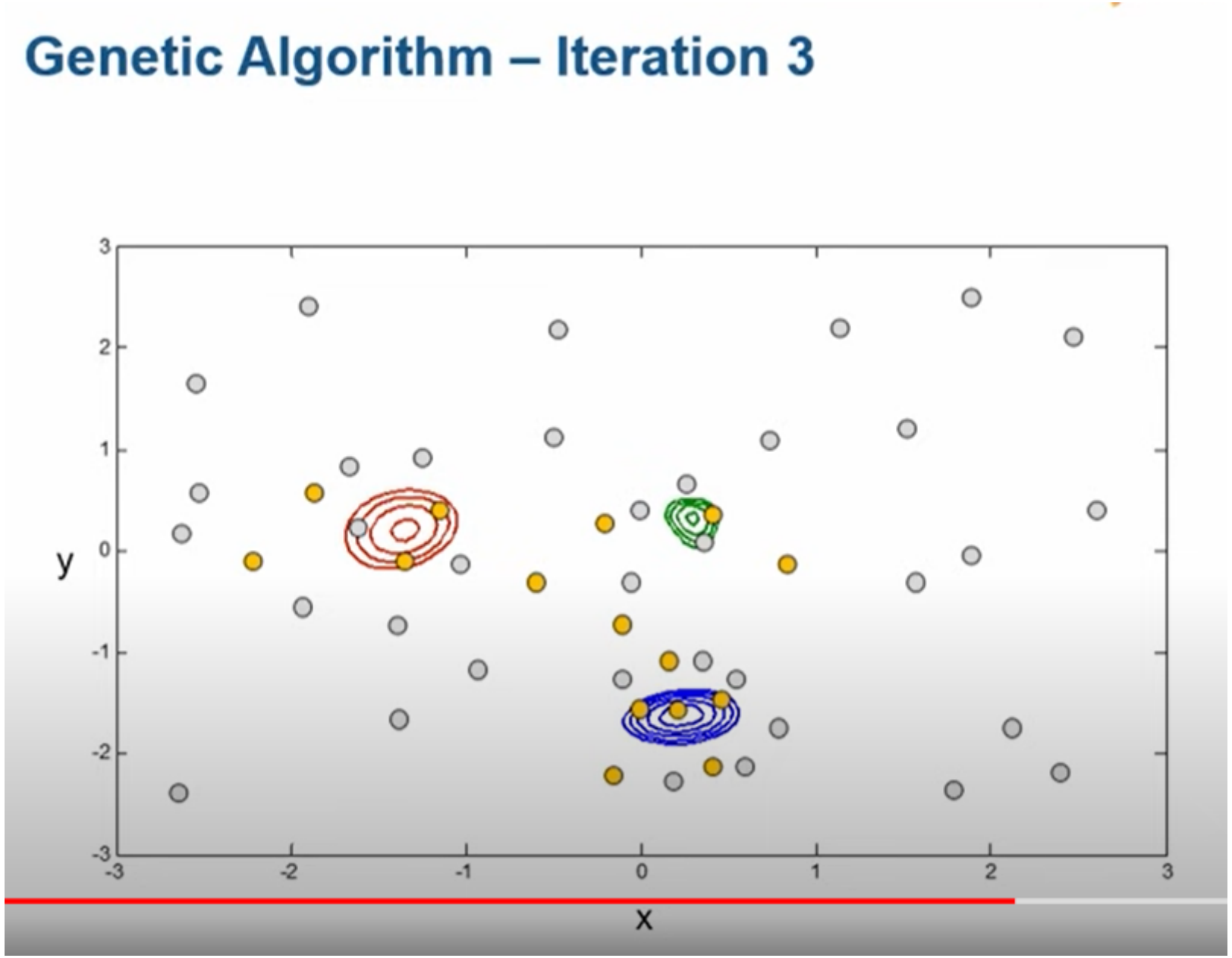
Generate new population and evaluate



Genetic Algorithm – Iteration 2

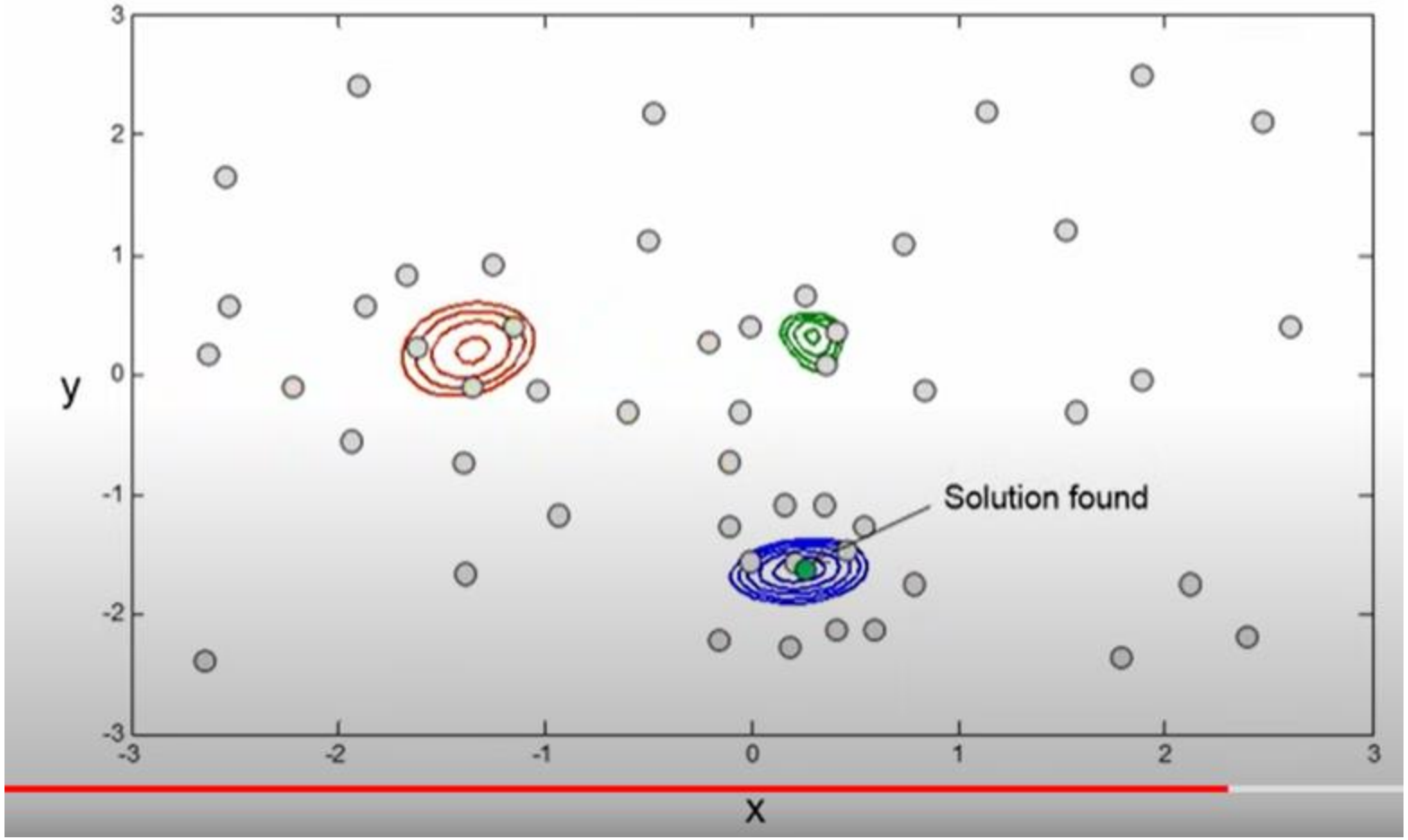


Genetic Algorithm – Iteration 3



Genetic Algorithm – Iteration N

Continue process until stopping criteria are met



Introduction

- Genetic Algorithm (GA) is a search-based optimization technique based on the principles of Genetics and Natural Selection.
- (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 (also called recombination)
- (GA)s are inspired by Darwin's theory about evolution-"survival of the fittest"

GAs were developed by John Holland and his students and colleagues at the University of Michigan, most notably David E. Goldberg and has since been tried on various optimization problems with a high degree of success

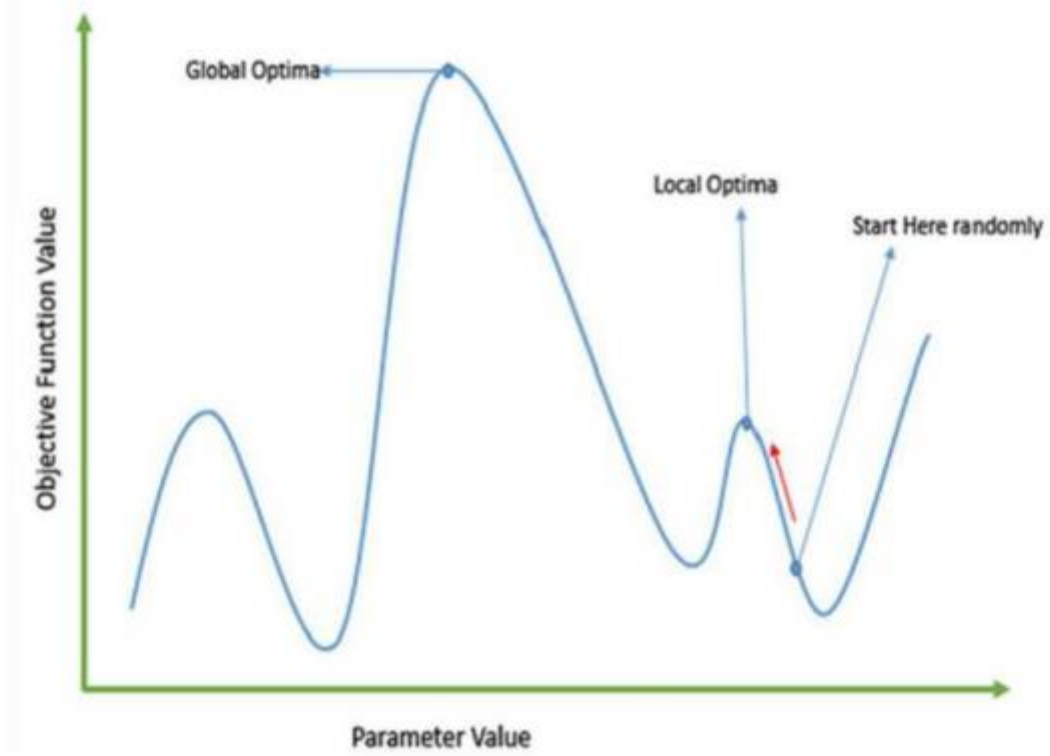
In GAs, we have a pool or a population of possible solutions to the given problem. These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations

Each individual (or candidate solution) is assigned a fitness value (based on its objective function value) and the fitter individuals are given a higher chance to mate and yield more "fitter" individuals

- **In this way we keep "evolving" better individuals or solutions over generations, till we reach a stopping criterion**
- **Genetic Algorithms are sufficiently randomized in nature, but they perform much better than random local search (in which we just try various random solutions, keeping track of the best so far), as they exploit historical information as well**

Why?

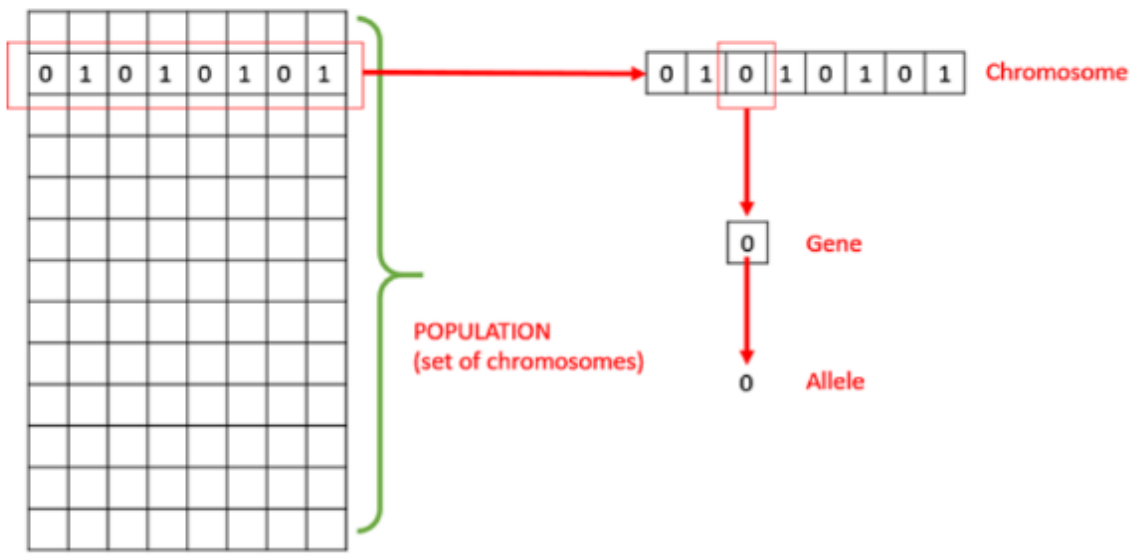
- **Solving Difficult problems**
- **Failure of Gradient Based Methods**



- **Getting a Good solution Fast**

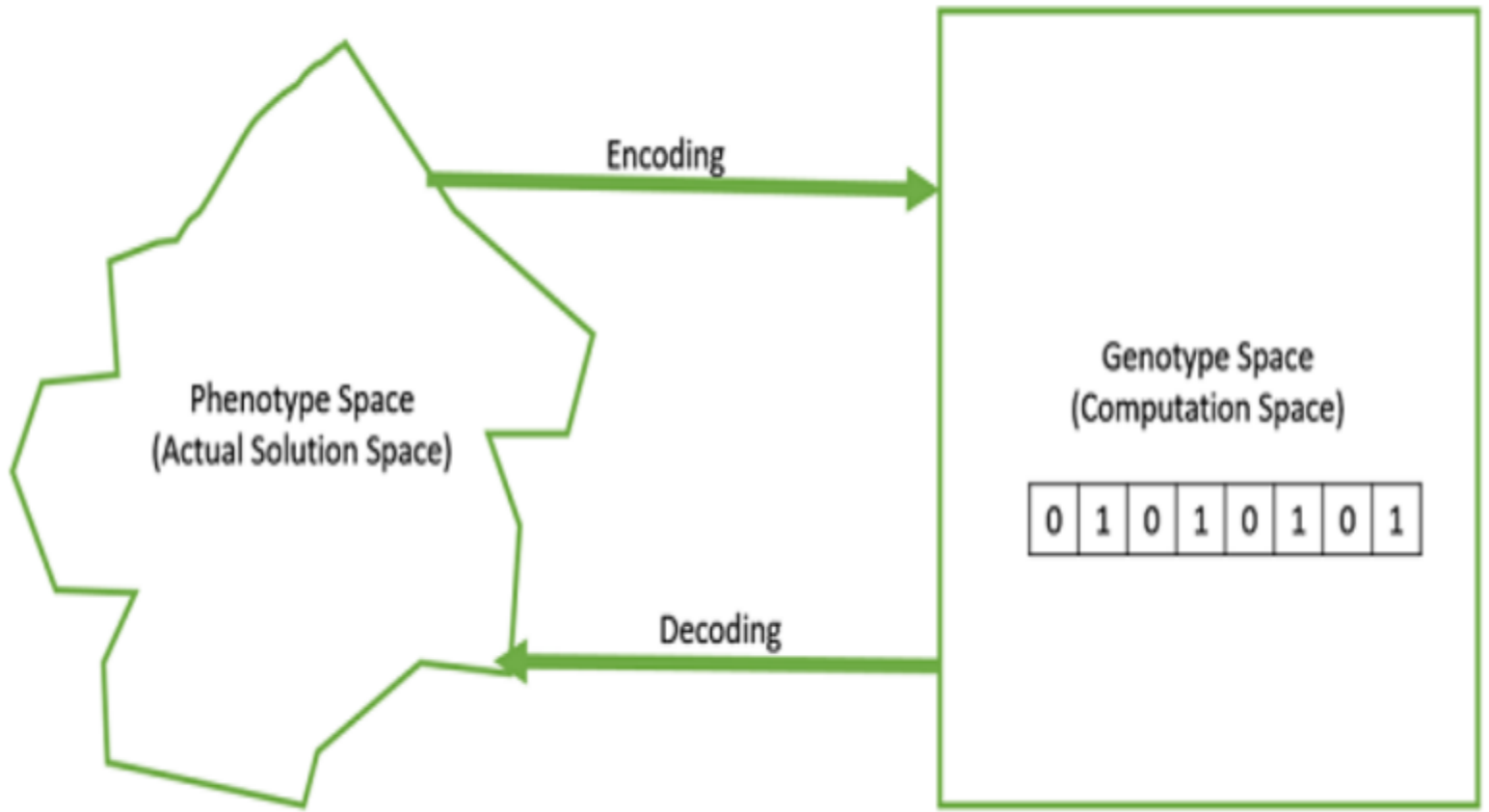
Basic Terminology

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



Basic Terminology

- **Genotype − Genotype is the 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 − 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.**


Encoding
Phenotype Space
(Actual Solution Space)
Genotype Space
(Computation Space)
0 1 0 1 0 1 0 1
Decoding

Example: 0-1 Knapsack problem

- There are n items, each item has its own cost (c_i) and weight (w_i).
- There is a knapsack of total capacity w.
- The problem is to take as much items as possible but not exceeding the capacity of the knapsack.

This is an optimization problem and can be better described as follows.

Maximize :

$$\sum_i c_i * w_i * x_i$$

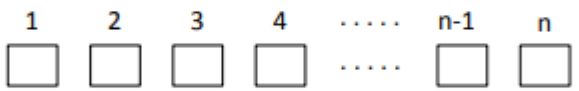
Subject to

$$\sum x_i * w_i \leq W$$

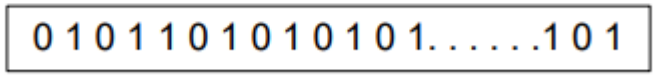
where $x_i \in [0 \cdots 1]$

The encoding for the 0-1 Knapsack, problem, in general, for *n* items set would look as follows.

Phenotype



Genotype.



A binary string of n-bits

- **Example 1 :**

  Minimize :

$$f(x) = \frac{x^2}{2} + \frac{125}{x}$$

  where $0 \leq x \leq 15$ and x is any discrete integer value.

Phenotype

x

Genotype

0 1 1 0 1

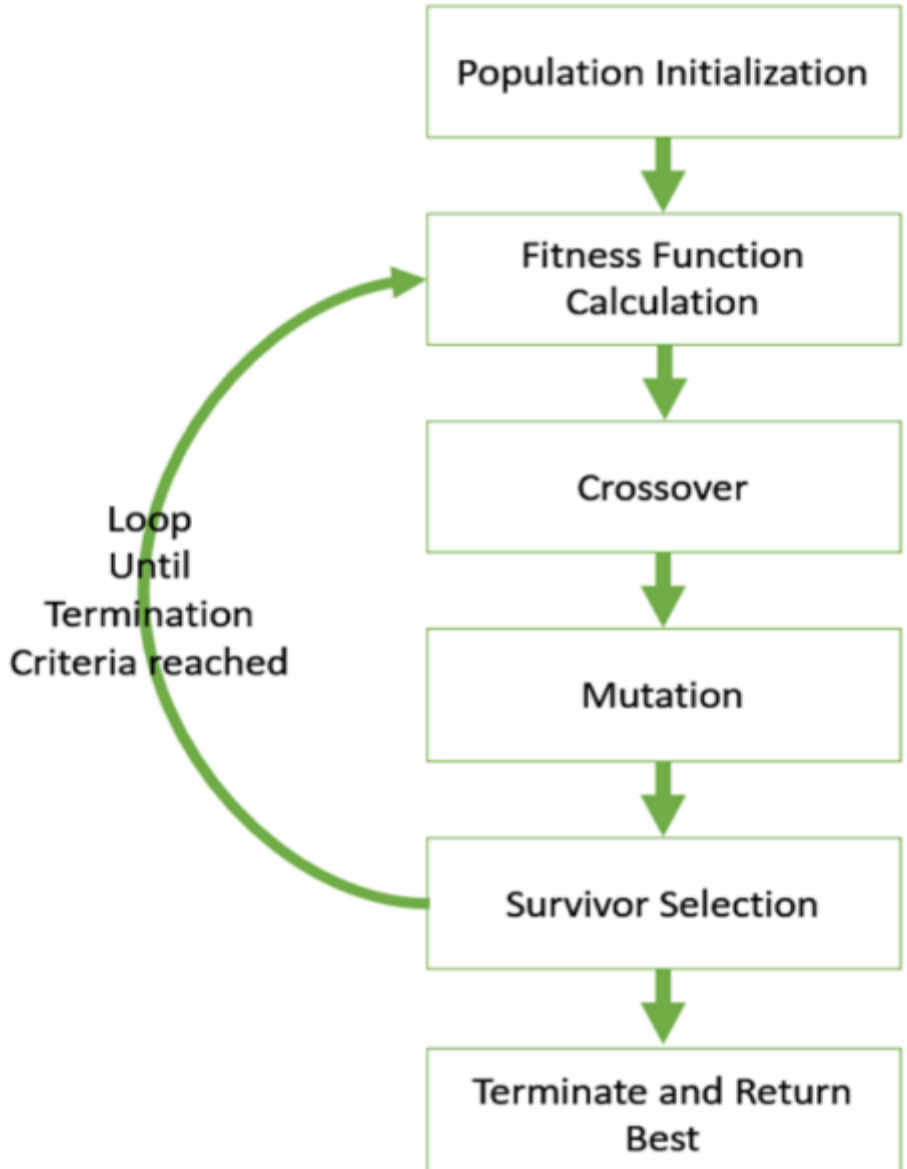
A binary string of 5-bits

- **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 selection, crossover, mutation**

GA Operators

In fact, a GA implementation involved with the realization of the following operations.

1. **Encoding:** How to represent a solution to fit with GA framework.
2. **Convergence:** How to decide the termination criterion.
3. **Mating pool:** How to generate next solutions.
4. **Fitness Evaluation:** How to evaluate a solution.
5. **Crossover:** How to make the diverse set of next solutions.
6. **Mutation:** To explore other solution(s).


Population Initialization
Fitness Function Calculation
Crossover
Mutation
Survivor Selection
Terminate and Return Best
Loop Until Termination Criteria reached

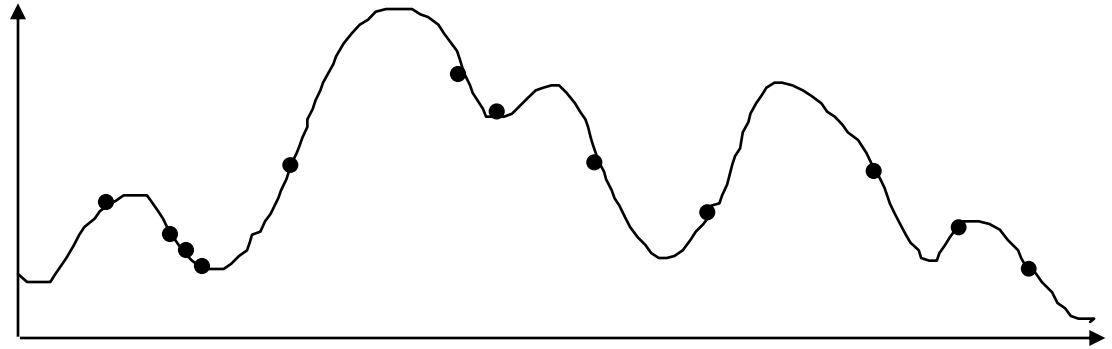
General Example:

- Suppose you want to be friends with the most intelligent student in the classroom.
- Finding the most intelligent student in the whole class:
  - One approach : Asking each and every student their CPIs.
  - Another approach : Randomly choose a group of 5 people. Find their fitness value – Here the fitness value is their CPI. Possibly the group has contacts of next 4 intelligent people. So the new offspring (Here friends to the initial 5 people) are fitter than the previous 5. Environment – (Most intelligent people group) So, with each generation the population improves and we get better results. Finally we find the most intelligent student in the classroom.

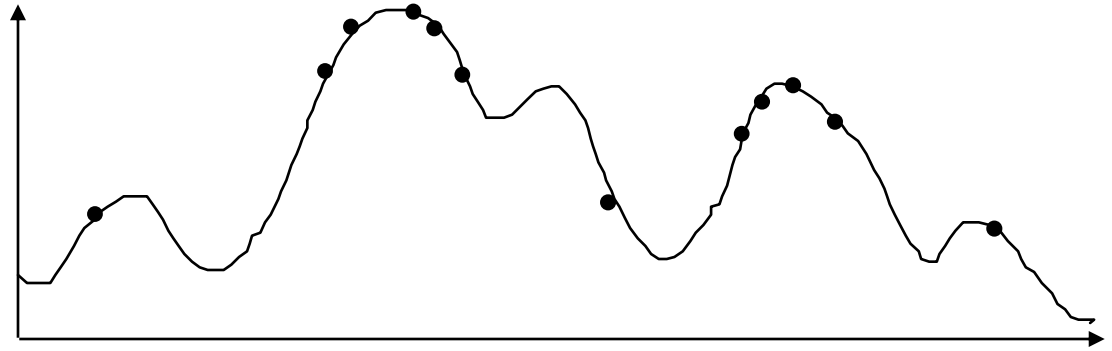
Simple Genetic Algorithm

```
{
    initialize population;
    evaluate population;
    while TerminationCriteriaNotSatisfied
    {
        select parents for reproduction;
        perform crossover and mutation;
        evaluate population;
    }
}
```

An Abstract Example



Distribution of Individuals in Generation 0



Distribution of Individuals in Generation N

Population Initialization

- There are two primary methods to initialize a population in a GA
  - Random Initialization – Populate the initial population with completely random solutions.
  - Heuristic initialization – Populate the initial population using a known heuristic for the problem.

Fitness

- A fitness score is given to each individual which shows the " ability of an individual to compete"
- Individual having better fitness score are given more chance to reproduce than others
- Individuals with better fitness scores are selected who mate and produce better offspring by combining chromosomes of parents

GA Operators

- **Methods of representation**
- **Methods of selection**
- **Methods of Reproduction**

Method of Representation – Encoding

There are many ways of encoding:

1. **Binary encoding:** Representing a gene in terms of bits (0s and 1s).
2. **Real value encoding:** Representing a gene in terms of values or symbols or string.
3. **Permutation (or Order) encoding:** Representing a sequence of elements)

Real value encoding

- The real-coded GA is most suitable for optimization in a continuous search space.
- Uses the direct representations of the design paparmeters
- Thus, avoids any intermediate encoding and decoding ste

Genotype :

x	y

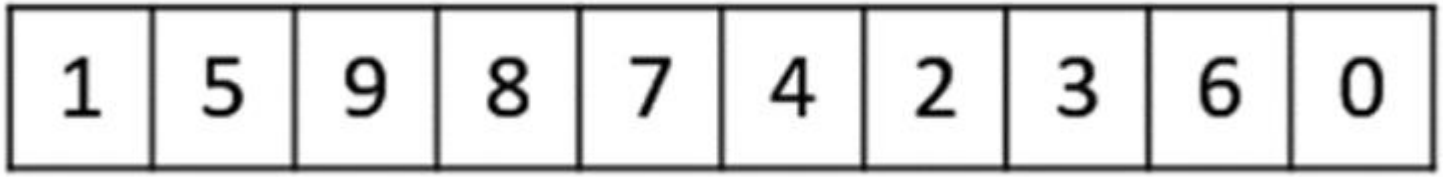
Phenotype :

5.28	-475.36

Real-value representation

Order Encoding

- In many problems, the solution is represented by an order of elements. In such cases permutation representation is the most suited.
- A classic example of this representation is the travelling salesman problem (TSP). In this the salesman has to take a tour of all the cities, visiting each city exactly once and come back to the starting city. The total distance of the tour has to be minimized. The solution to this TSP is naturally an ordering or permutation of all the cities and therefore using a permutation representation makes sense for this problem.
-

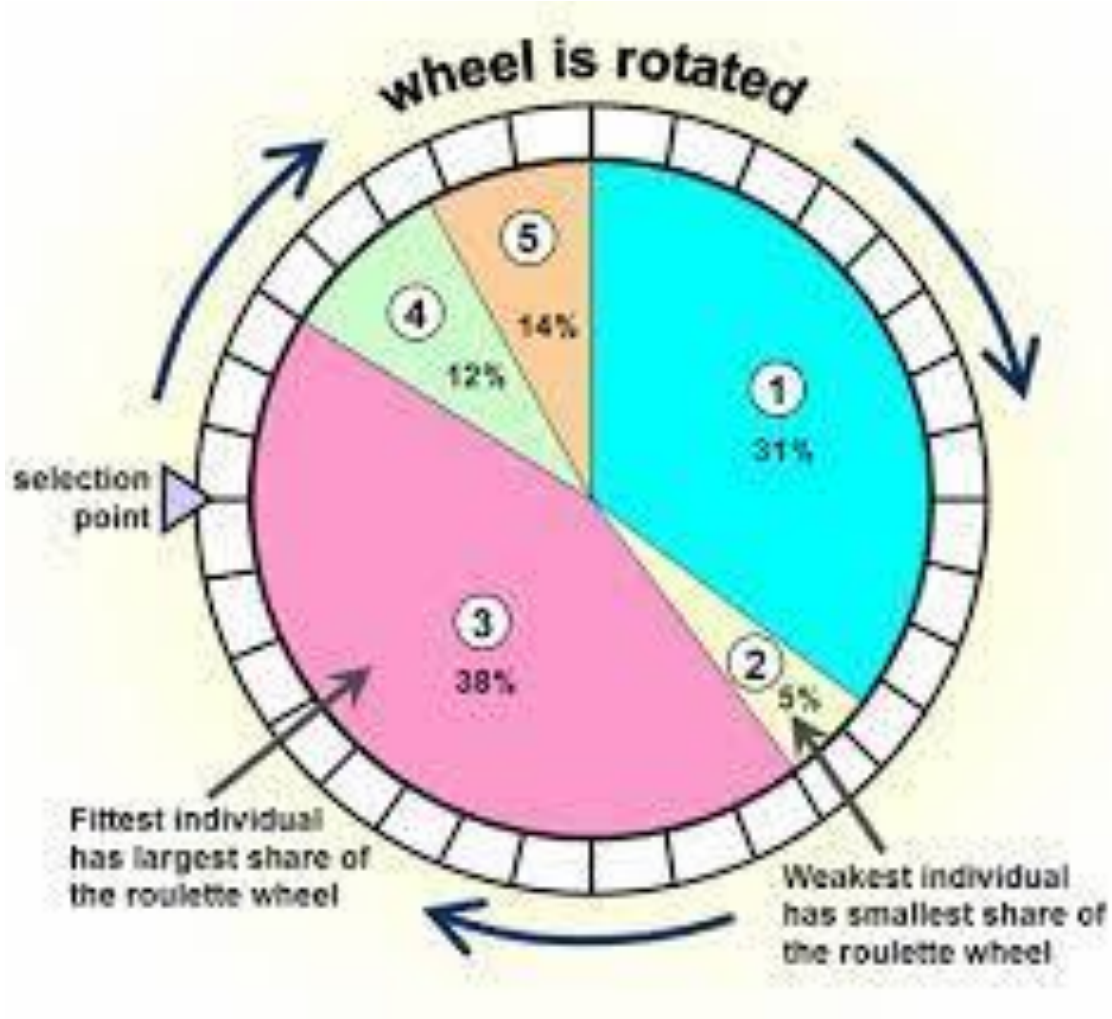


Methods of Selection

There are many different strategies to select the individuals to be copied over into the next generation

Methods of Selection

- *Roulette-wheel selection*.
- *Elitist selection*.
- *Rank selection*.
- Tournament Selection.


wheel is rotated
selection point
1
31%
2
5%
3
38%
4
12%
5
14%
Fittest individual has largest share of the roulette wheel
Weakest individual has smallest share of the roulette wheel

Roulette wheel selection

- Conceptually, this can be represented as a game of roulette - each individual gets a slice of the wheel, but more fit ones get larger slices than less fit ones.

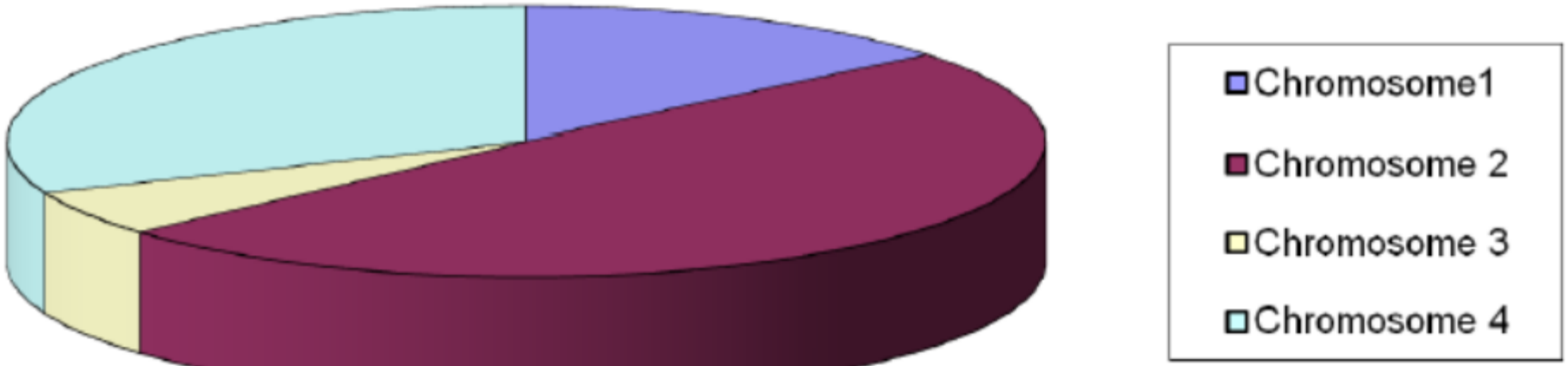
Roulette wheel selection

No.	String	Fitness	% Of Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0

- The top surface area of the wheel is divided into N parts in proportion to the fitness values $f_1, f_2, f_3 \cdots f_N$.
- The wheel is rotated in a particular direction (either clockwise or anticlockwise) and a fixed pointer is used to indicate the winning area, when it stops rotation.
- A particular sub-area representing a GA-Solution is selected to be winner probabilistically and the probability that the $i-th$ area will be declared as

$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}$$

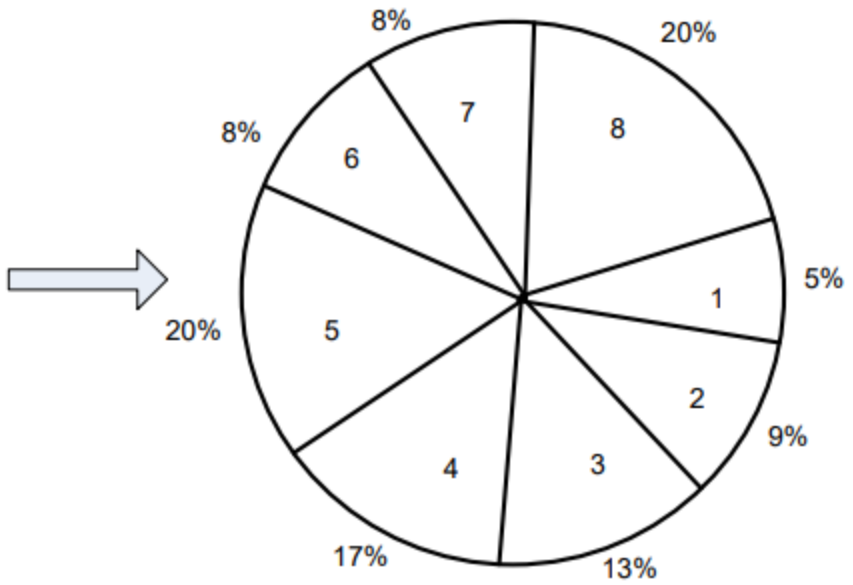
- In other words, the individual having higher fitness value is likely to be selected more.



- **Let the points at which it has stopped are: 2, 3, 4 and 1**
- **Then the four chromosomes are selected based on the fitness probability and these four chromosomes are called the mating pool**

Roulette-Wheel selection mechanism: An Example

Individual	Fitness value	p_i
1	1.01	**0.05**
2	2.11	**0.09**
3	3.11	**0.13**
4	4.01	**0.17**
5	4.66	**0.20**
6	1.91	**0.08**
7	1.93	**0.08**
8	4.51	**0.20**



Roulette-Wheel selection : Implementation

Input: A Population of size N with their fitness values
Output: A mating pool of size N_p

Steps:

1. Compute $p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}, \forall i = 1, 2 \cdots N$
2. Calculate the cumulative probability for each of the individual starting from the top of the list, that is $P_i = \sum_{j=1}^{i} p_j$, for all $j = 1, 2 \cdots N$
3. Generate a random number say r between 0 and 1.
4. Select the j-th individual such that $P_{j-1} < r \leq P_j$
5. Repeat Step 3-4 to select N_p individuals.
6. End

Roulette-Wheel selection: Example

The probability that i-th individual will be pointed is

$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}$$

Example:

Individual	p_i	P_i	r	T
1	0.05	0.05	0.26	I
2	0.09	0.14	0.04	I
3	0.13	0.27	0.48	II
4	0.17	0.44	0.43	I
5	0.20	0.64	0.09	II
6	0.08	0.72	0.30	
7	0.08	0.80	0.61	
8	0.20	1.0	0.89	I

p_i = Probability of an individual

r = Random Number between 0..1

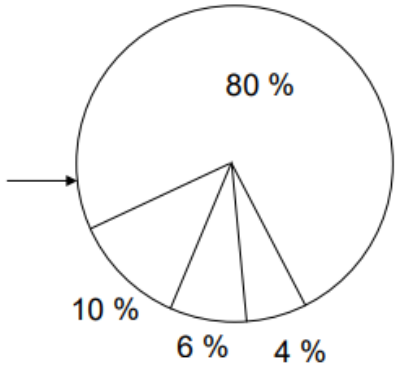
P_i = Cumulative Probability

T=Tally count of selection

Issue

The limitations in the Roulette-Wheel selection scheme can be better illustrated with the following figure.

Individual (i)	Fitness (fi)	RW (Area)
1	0.4	80 %
2	0.05	10 %
3	0.03	6 %
4	0.02	4 %

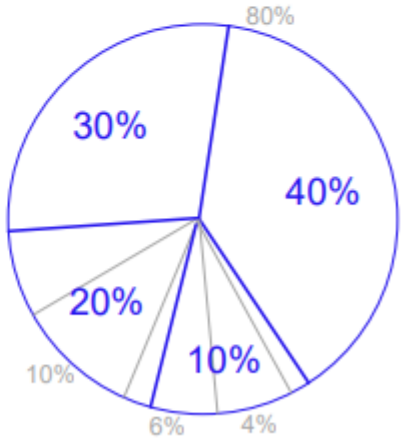


The observation is that the individual with higher fitness values will guard the other to be selected for mating. This leads to a lesser diversity and hence fewer scope toward exploring the alternative solution and also premature convergence or early convergence with local optimal solution.

Rank Based Selection Method

- Continuing with the population of 4 individuals with fitness values $f_1 = 0.40, f_2 = 0.05, f_3 = 0.03$ and $f_4 = 0.02$.
- Their proportionate area on the wheel are: 80%, 10%, 6% and 4%
- Their ranks are shown in the following figure.

Individual (i)	Fitness (fi)	RW (Area)	Rank	RS (Area)
1	0.4	80 %	4	40 %
2	0.05	10 %	3	30 %
3	0.03	6 %	2	20 %
4	0.02	4 %	1	10 %



Input: A population of size N with their fitness values
Output: A mating pool of size N_p.

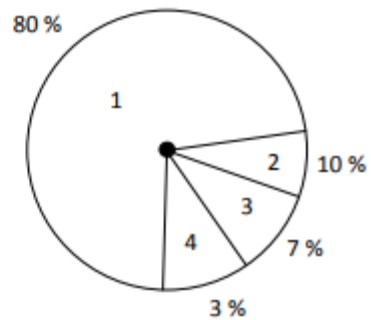
Steps:

1. Arrange all individuals in ascending order of their fitness value.
2. Rank the individuals according to their position in the order, that is, the worst will have rank 1, the next rank 2 and best will have rank N.
3. Apply the Roulette-Wheel selection but based on their assigned ranks. For example, the probability p_i of the i-th individual would be

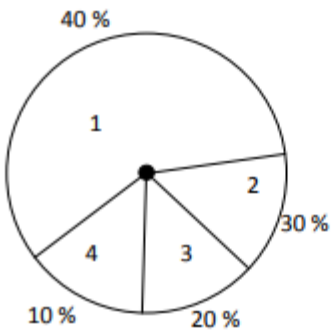
$$p_i = \frac{r_i}{\sum_{j=1}^{i} r_j}$$

4. Stop

Individual	% Area	f_i	Rank (r_i)	% Area
1	80 %	0.4	4	40 %
2	10 %	0.05	3	30 %
3	7 %	0.03	2	20 %
4	4 %	0.02	1	10 %



Roulette-Wheel based on proportionate-based selection



Roulette-Wheel based on ordinal-based selection

Effectiveness of any selection scheme

- Population diversity:
  - This is similar to the concept of exploration. The population diversity means that the genes from the already discovered good individuals are exploited while permitting the new area of search space continue to be explored.
  - Selection pressure:
    - This is similar to the concept of exploitation. It is defined as the degree to which the better individuals are favoured.

- If Selection Pressure is High – Less exploration
- If Selection Pressure is low – More time to converge

Other selection methods

- *Elitist selection*:

  Chose only the most fit members of each generation.
- *Cutoff selection*:

  Select only those that are above a certain cutoff for the target function.

Methods of Reproduction

- There are primary methods:
  - *Crossover*
  - *Mutation*

Methods of Reproduction: Crossover

- Two parents produce two offspring
- Two options:

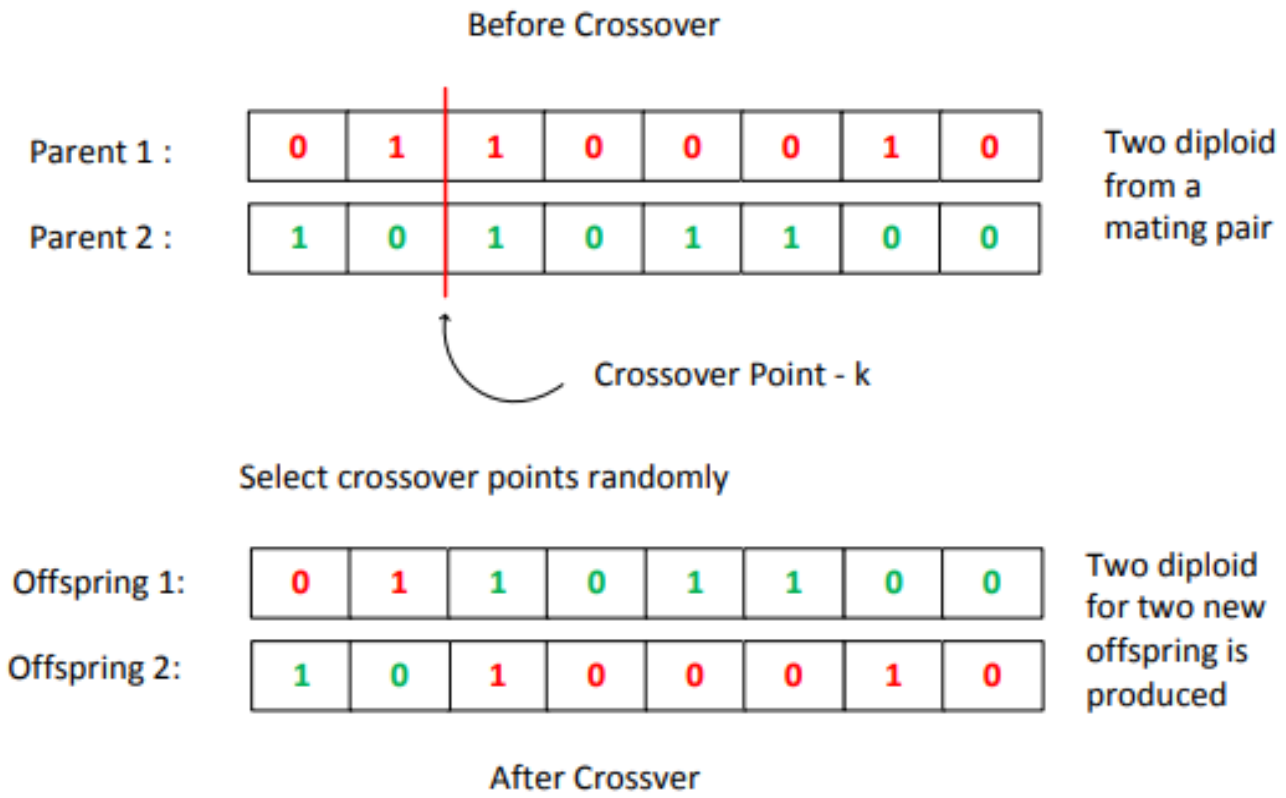
  1.The chromosomes of the two parents are copied to the next generation

  2.The two parents are randomly recombined (crossed-over) to form new offsprings

Several possible crossover strategies

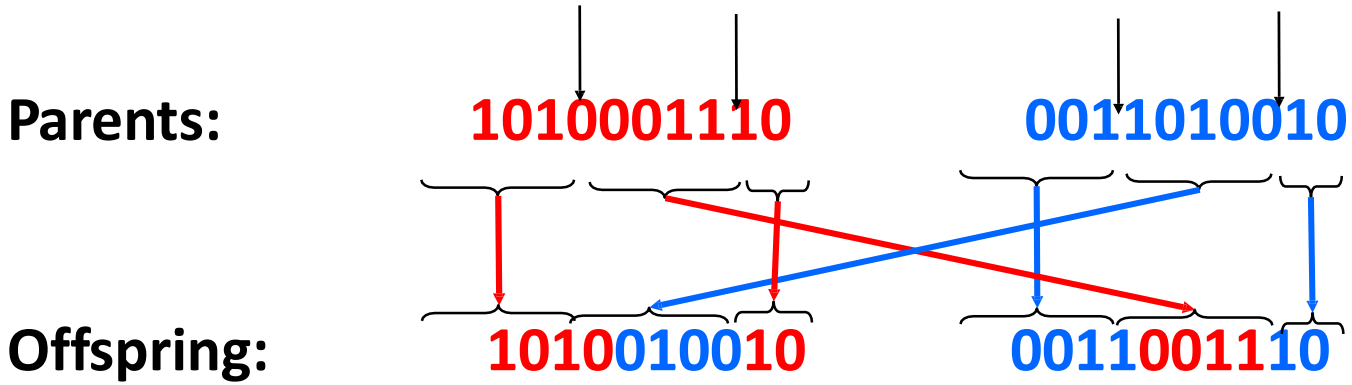
- Randomly select a single point for a crossover
- Multi point crossover
- Uniform crossover

Single Point Crossover



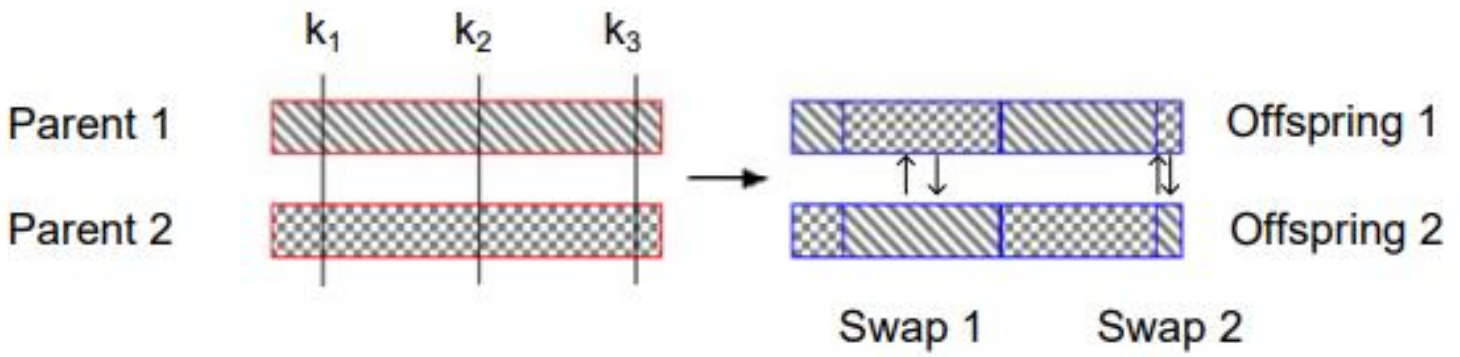
Two-point crossover

- In this scheme, we select two different crossover points k1 and k2 lying between 1 and L at random such that k1 != k2.
- The middle parts are swapped between the two strings. Alternatively, left and right parts also can be swapped.



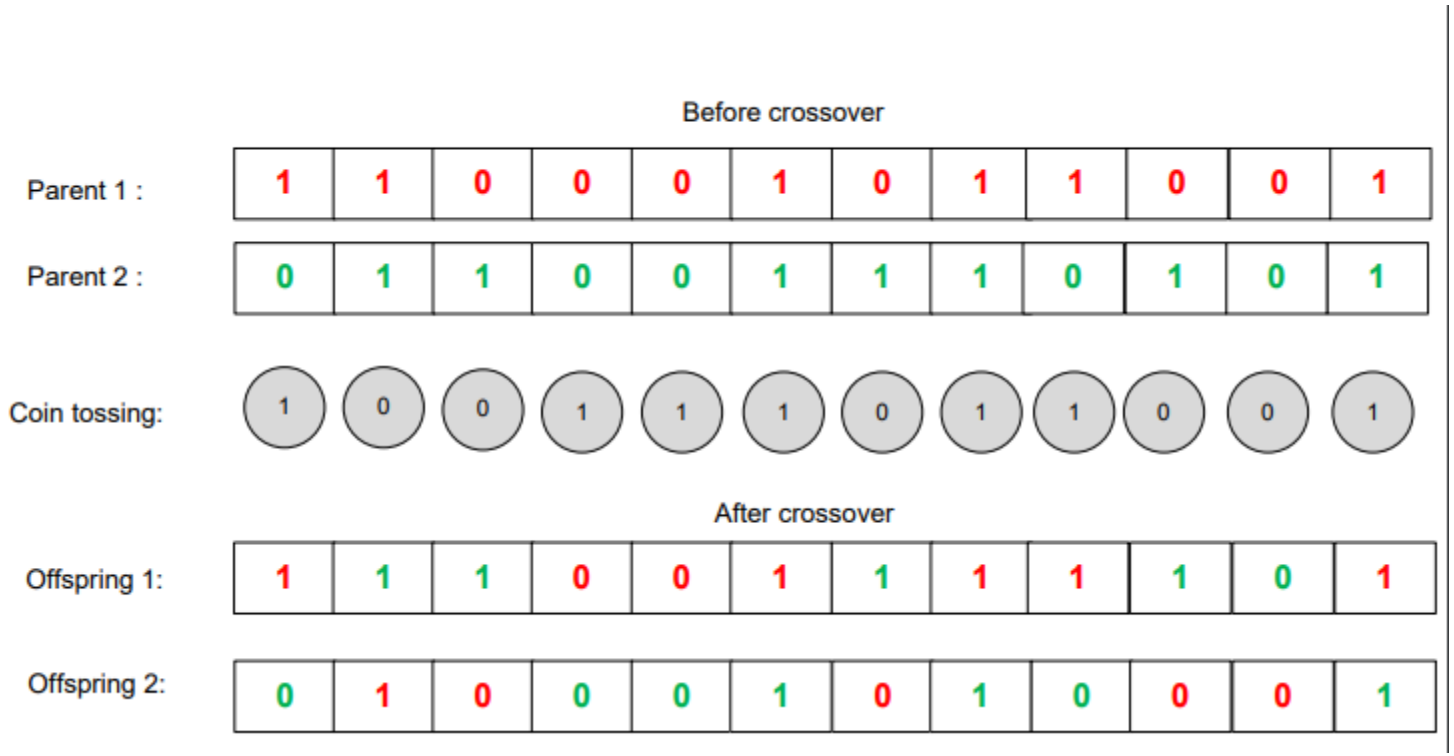
Multi Point Crossover

1. In case of multi-point crossover, a number of crossover points are selected along the length of the string, at random.
2. The bits lying between alternate pairs of sites are then swapped.



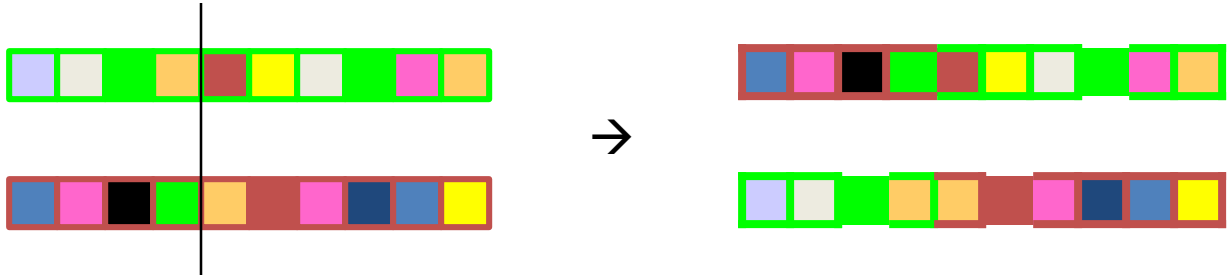
Uniform Crossover

- Uniform crossover is a more general version of the multi-point crossover.
- In this scheme, at each bit position of the parent string, we toss a coin (with a certain probability p_s) to determine whether there will be swap of the bits or not.
- The two bits are then swapped or remain unaltered, accordingly.

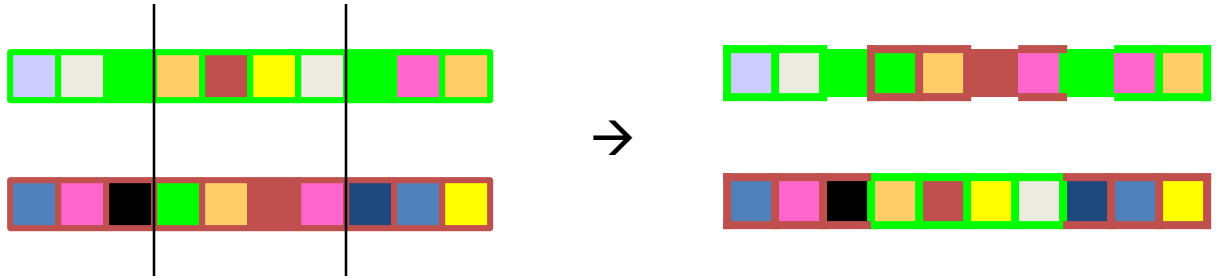

Before crossover
Parent 1 :
1 1 0 0 0 1 0 1 1 0 0 1
Parent 2 :
0 1 1 0 0 1 1 1 0 1 0 1
Coin tossing:
1 0 0 1 1 1 0 1 1 0 0 1
After crossover
Offspring 1:
1 1 1 0 0 1 1 1 1 1 0 1
Offspring 2:
0 1 0 0 0 1 0 1 0 0 0 1

Crossover

- Single point crossover



- Two point crossover (Multi point crossover)



Mutation

- Mutation is a genetic operator used to maintain genetic diversity from one generation of population of chromosomes to the next.
- Mutation occurs during evolution according to a user defined mutation probability, usually set to fairly low value, say 0.01. For example, with a mutation rate of 0.01, it might be expected that one gene in a chromosome of 100 genes might be reversed.
- Mutation simply involves reversing the value of a bit in a chromosome. This can result in entirely new gene values being added to the gene pool.
- With the new gene values, the genetic algorithm may be able to arrive at better solution than was previously possible

Methods of Reproduction: Mutations

- **Mutation helps to prevent the population from stagnating at any <u>local optima</u>**
- **Consider the two original off-springs selected for mutation**

Original offspring 1	1	1	0	1	1	1	1	0	0	0	0	1	1	1	1	0
Original offspring 2	1	1	0	1	1	0	0	1	0	0	1	1	0	1	1	0

- **Invert the value of the chosen gene as 0 to 1 and 1 to 0**
- **The Mutated Off-spring produced are:**

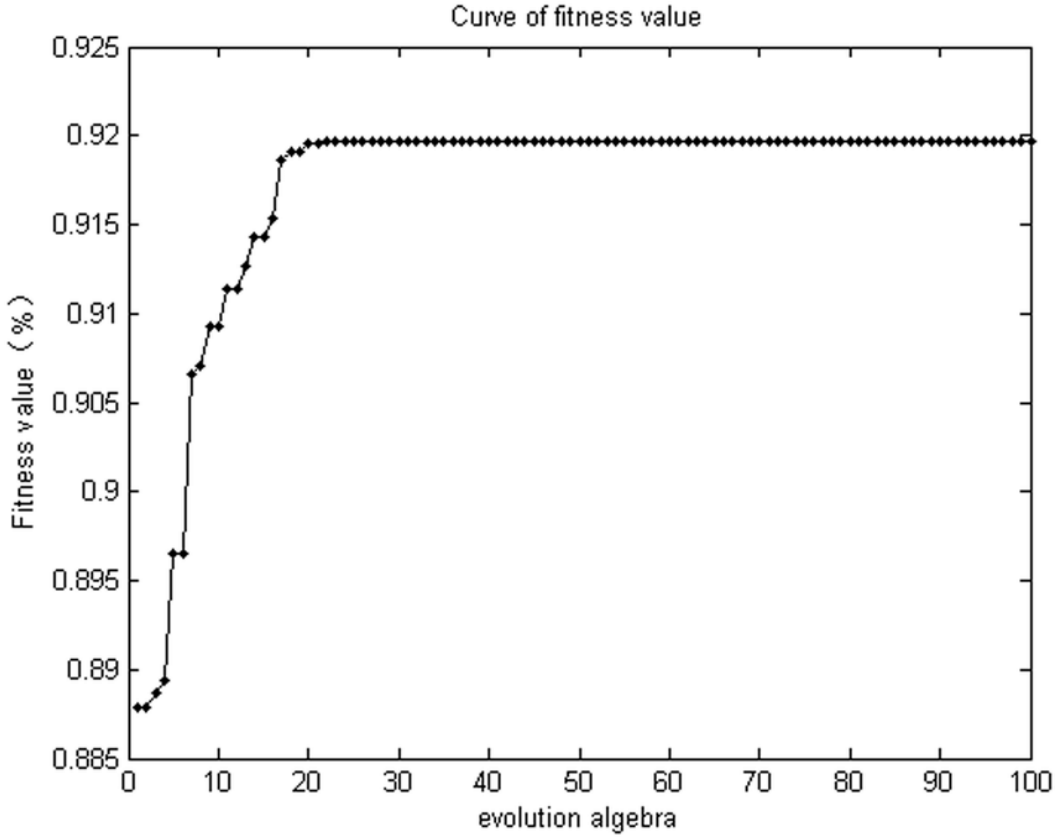
Mutated offspring 1	1	1	0	**0**	1	1	1	0	0	0	0	1	1	1	1	0
Mutated offspring 2	1	1	0	1	1	0	**1**	1	0	0	1	1	0	1	**0**	0

Termination Criteria

- Two ways in which a run of a genetic algorithm is terminated.
  - Limit is put on the number of generations, after which the run is considered to have finished.
  - Run can stop when a particular solution has been reached, or when the highest fitness level in the population has reached a particular value

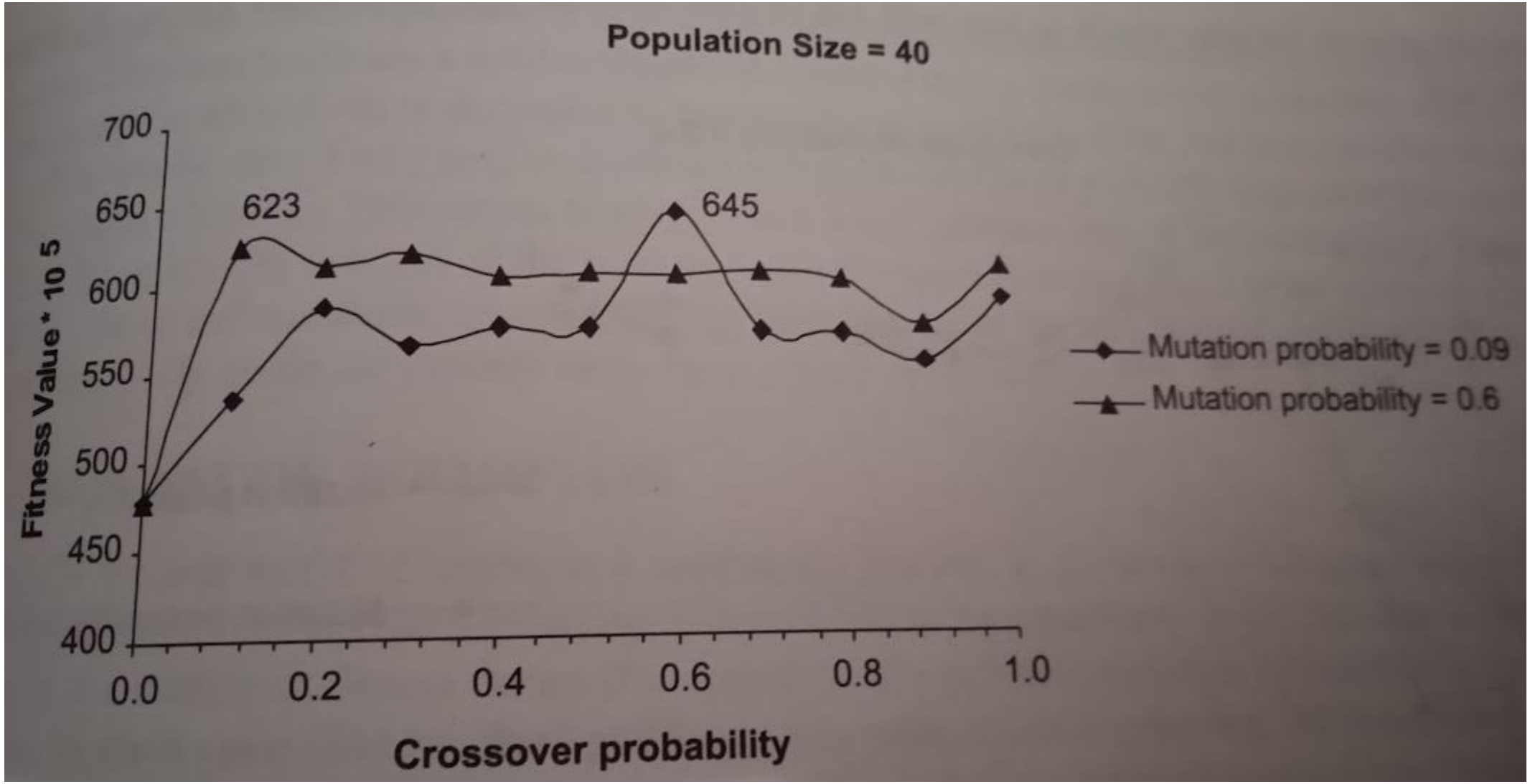
Mutation and Crossover Probability

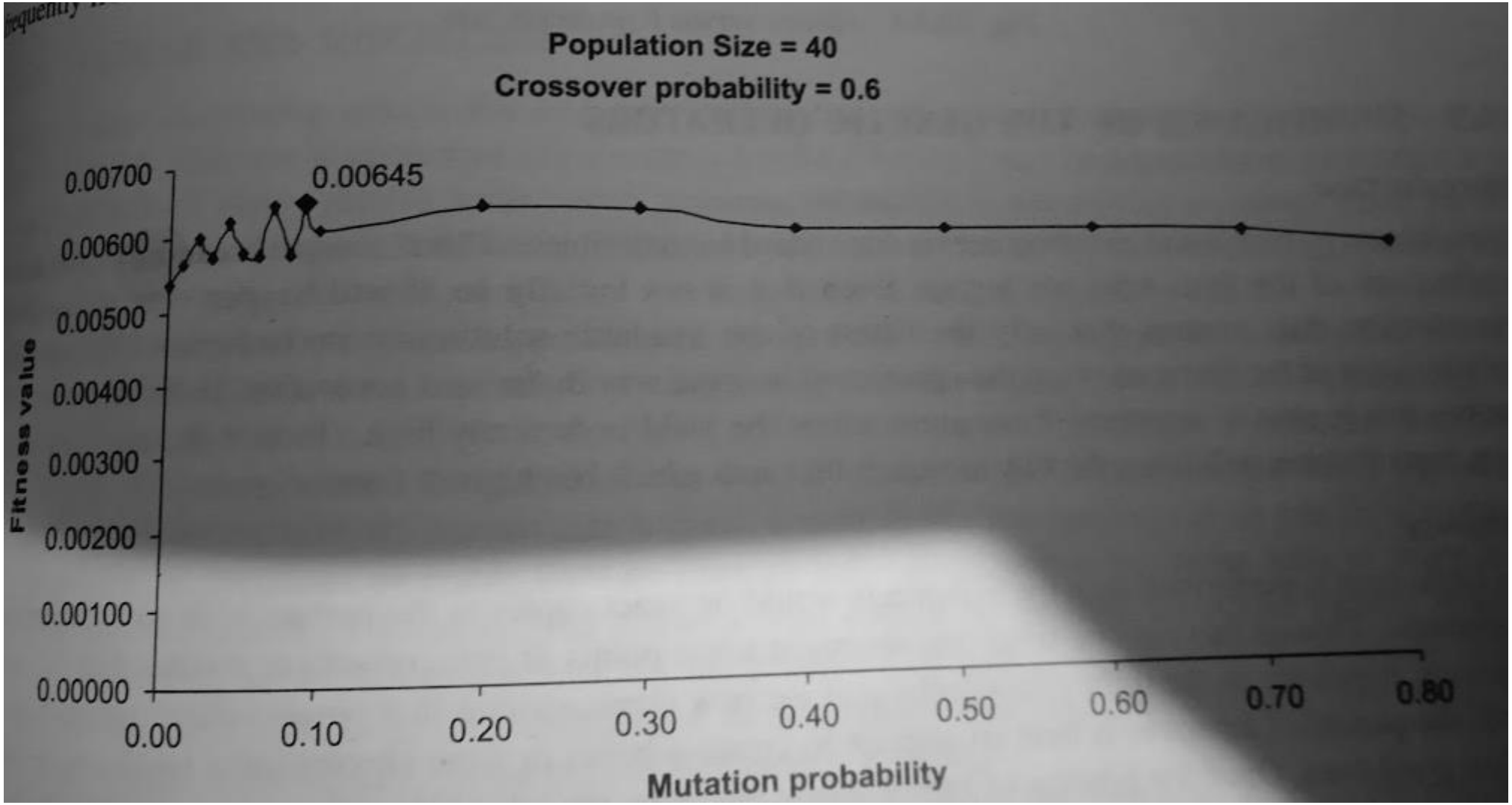
- Skill based employee allocation : To achieve global minimum – it is necessary that GA move in every possible way.



- Fitness saturates after a certain number of generations and shows no change after that. This means that for a particular set(Population, Crossover probability = 0.6 and mutation probability=0.09) up the GA levels off after certain number of generations yielding a global maximum value.

Fitness vs Crossover Probability




Population Size = 40
Crossover probability = 0.6
0.00645
0.00700
0.00600
0.00500
0.00400
0.00300
0.00200
0.00100
0.00000
Fitness value
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
Mutation probability

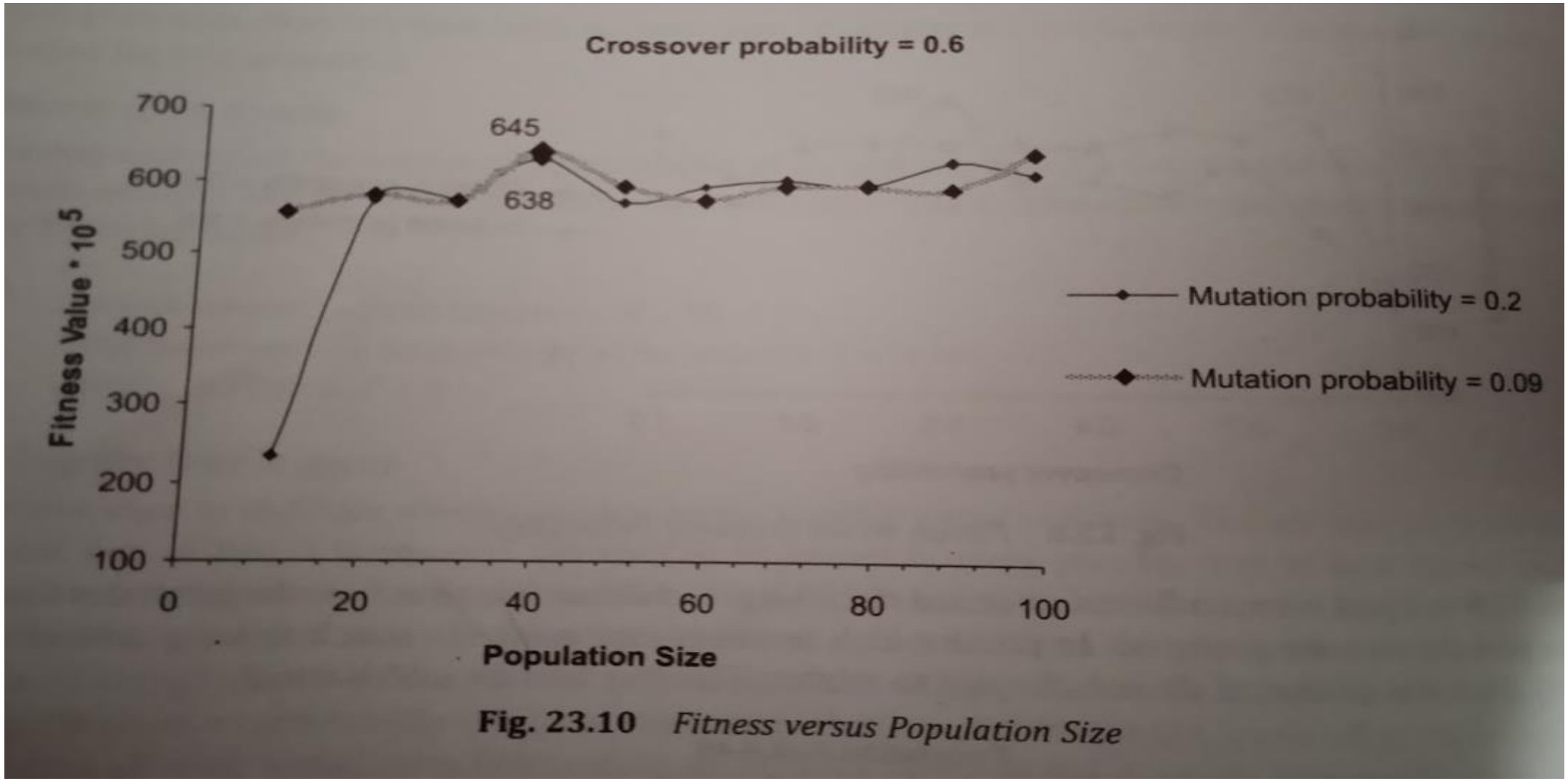


Fig. 23.10 *Fitness versus Population Size*

Issues for GA Practitioners

- Choosing basic implementation issues:
  - representation
  - population size, mutation rate, ...
  - selection, deletion policies
  - crossover, mutation operators
- Termination Criteria
- Performance, scalability
- Solution is only as good as the evaluation function (often hardest part)