

# Genetic Algorithm

**Made By**  
-Garima Singhal





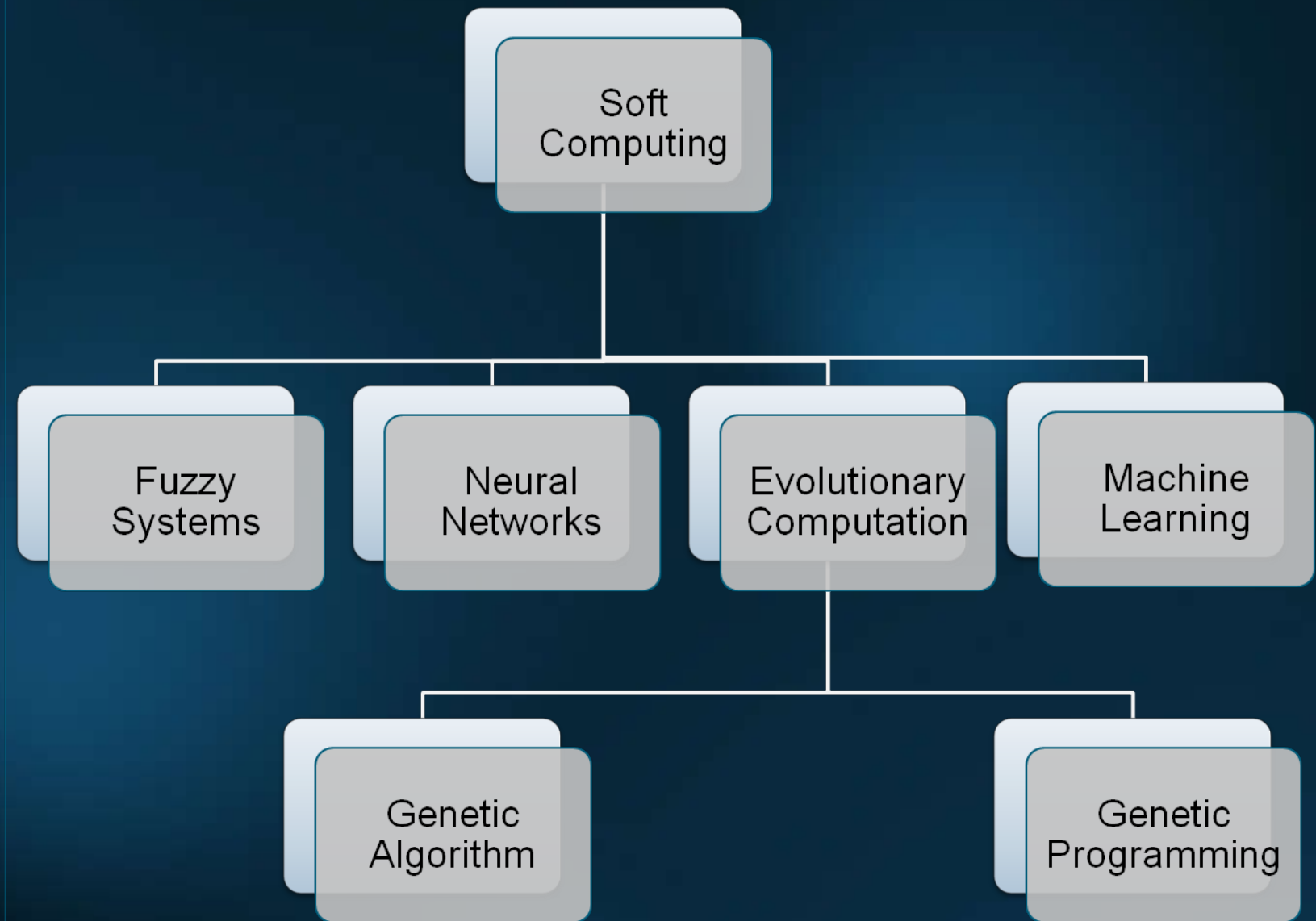
# Introduction

- One of the central challenges of computer science is to get a computer to do what needs to be done, without telling it how to do it.

- 
- Before looking forward to what actually is **GENETIC PROGRAMMING** we shall first get acquainted with **Soft Computing**.



# Soft Computing





# Soft Computing


- It is another field of computer science.
- It is the fusion of methodologies that were designed to model and enable solutions to real world problems, which are not modeled, or too difficult to model, mathematically.
- It is associated with fuzzy, complex and dynamic system with uncertain parameters.
- Quality of Service Networking is an example of the class of complex, fuzzy and dynamic systems with uncertain parameters, which soft computing is intended to model and compute.





# Difference between Soft computing And Conventional Computing

Hard Computing(Conventional)	Soft Computing
Requires precisely stated analytical model and a lot of computation time.	Tolerant of imprecision, uncertainty and approximation.
Based on binary logic, numerical analysis and crisp software.	Based on fuzzy logic, neural networks and probabilistic reasoning.
Requires programs to be written.	Can evolve its own programs.
Deterministic.	Incorporates stochasticity.
Strictly sequential.	Allows parallel computations.



# Current Applications of Soft Computing include

- Handwriting Recognition
- Image Processing and Data Compression
- Automotive Systems and Manufacturing
- Decision –Support Systems
- Neurofuzzy Systems
- Fuzzy Logic Control





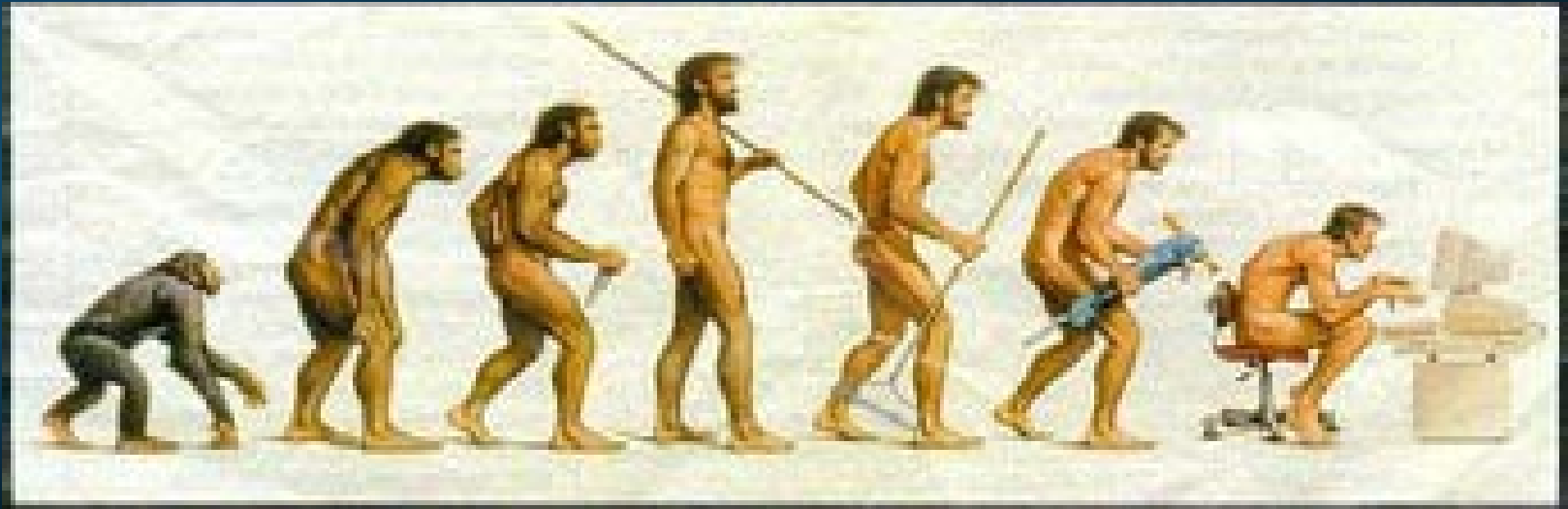
# Evolutionary Computation

- Evolutionary computation simulates evolution on a computer. The result of such a simulation is a series of optimization algorithms, usually based on a simple set of rules. Optimization iteratively improves the quality of solutions until an optimal, or at least feasible, solution is found.

# Contd...

- Evolutionary Computations can be studied under 2 categories:
  - Genetic Algorithm
  - Genetic programming

# Genetic Algorithm



Search techniques

Calculus-based techniques

Stochastic random search techniques

Enumerative techniques

Direct methods

Indirect methods

Evolutionary algorithm

Simulated annealing

Dynamic programming

Fibonacci

Newton

Evolutionary strategies

Genetic algorithm

Parallel

Sequential

Centralize

Distribute

Ready-state

Iteration



# Genetic Algorithm Terminologies

Now, before we start, we should understand some key terms :

<b>Individual</b>	Any possible solution
<b>Population</b>	Group of all <i>individuals</i>
<b>Search Space</b>	All possible solutions to the problem
<b>Chromosome</b>	Blueprint for an <i>individual</i>
<b>Trait</b>	Possible aspect of an <i>individual</i>
<b>Allele</b>	Possible settings for a <i>trait</i>
<b>Locus</b>	The position of a <i>gene</i> on the <i>chromosome</i>
<b>Genome</b>	Collection of all <i>chromosomes</i> for an <i>individual</i>





# Premise

- Evolution works biologically so maybe it will work with simulated environments.
- Here each possible solution (good or bad) is represented by a chromosome (i.e. a pattern of bits which is sort of synonymous to DNA)
- Determine the better solutions
- Mate these solutions to produce new solutions which are (hopefully!) occasionally better than the parents.
- Do this for many generations.



# Outline of the Genetic Algorithm

- Randomly generate a set of possible solutions to a problem, representing each as a fixed length character string
- Test each possible solution against the problem using a fitness function to evaluate each solution
- Keep the best solutions, and use them to generate new possible solutions
- Repeat the previous two steps until either an acceptable solution is found, or until the algorithm has iterated through a given number of cycles (generations)



# Basic Operators of Genetic Algorithm


- **Reproduction:** It is usually the first operator applied on population. Chromosomes are selected from the population of parents to cross over and produce offspring. It is based on Darwin's evolution theory of "Survival of the fittest". Therefore, this operator is also known as '**Selection Operator**'.
- **Cross Over:** After reproduction phase, population is enriched with better individuals. It makes clones of good strings but doesnot create new ones. Cross over operator is applied to the mating pool with a hope that it would create better strings.
- **Mutation:** After cross over, the strings are subjected to mutation. Mutation of a bit involves flipping it,changing 0 to 1 and vice-versa.



# Basic genetic algorithms

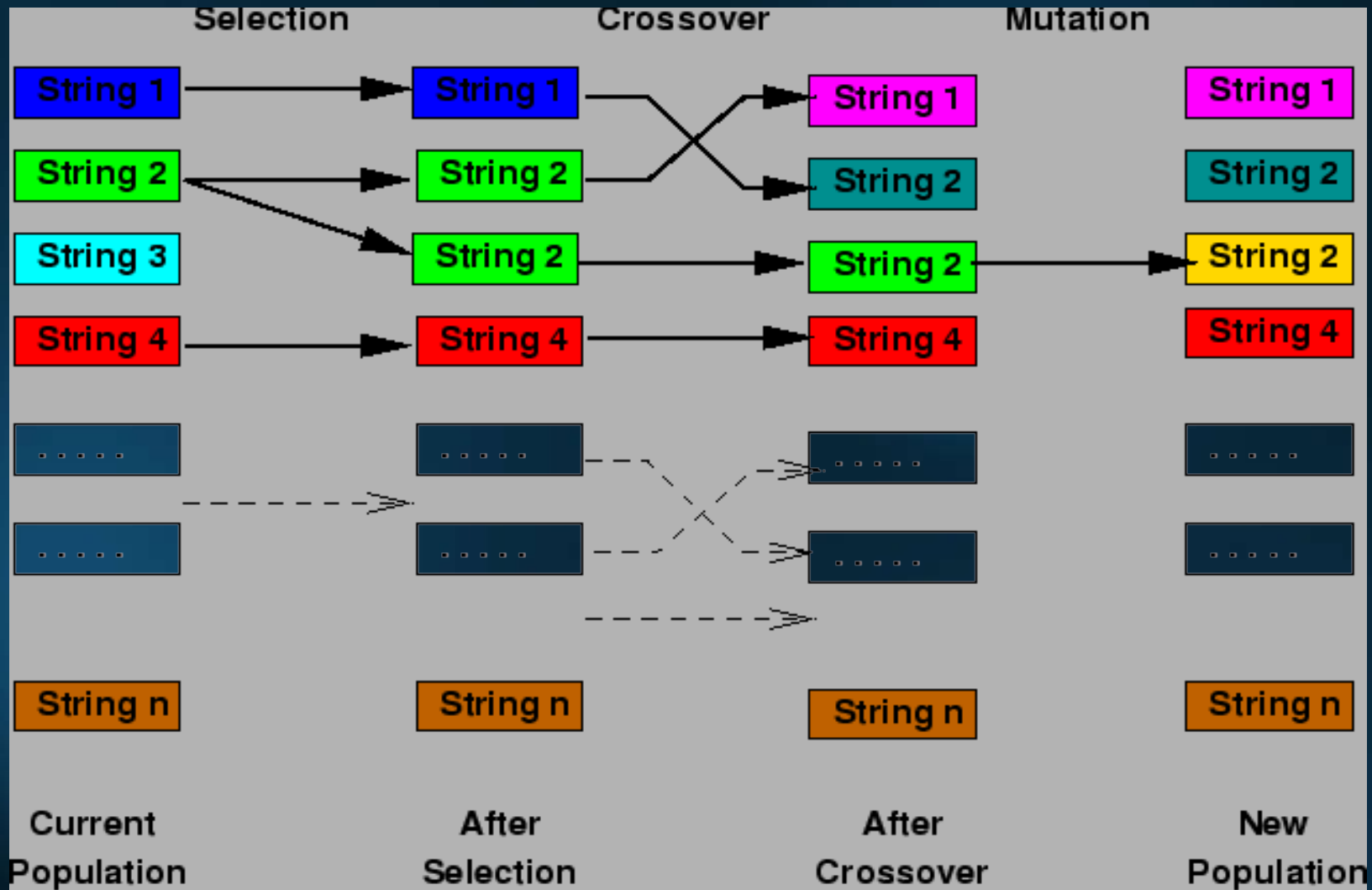
- **Step 1:** Represent the problem variable domain as a chromosome of a fixed length, choose the size of a chromosome population  $N$ , the crossover probability  $p_c$  and the mutation probability  $p_m$ .
- **Step 2:** Define a fitness function to measure the performance, or fitness, of an individual chromosome in the problem domain. The fitness function establishes the basis for selecting chromosomes that will be mated during reproduction.
- **Step 3:** Randomly generate an initial population of chromosomes of size  $N$ :  
 $x_1, x_2, \dots, x_N$
- **Step 4:** Calculate the fitness of each individual chromosome:  
 $f(x_1), f(x_2), \dots, f(x_N)$



- 
- **Step 5:** Select a pair of chromosomes for mating from the current population. Parent`chromosomes are selected with a probability related to their fitness.
  - **Step 6:** Create a pair of offspring chromosomes by applying the genetic operators – **crossover** and **mutation**.
  - **Step 7:** Place the created offspring chromosomes in the new population.
  - **Step 8:** Repeat *Step 5* until the size of the new chromosome population becomes equal to the size of the initial population,  $N$ .
  - **Step 9:** Replace the initial (parent) chromosom population with the new (offspring) population.
  - **Step 10:** Go to *Step 4*, and repeat the process until the termination criterion is satisfied.



# Working Principle of Genetic Algorithm






# Genetic Algorithms: Case Study

- A simple example will help us to understand how a GA works. 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:

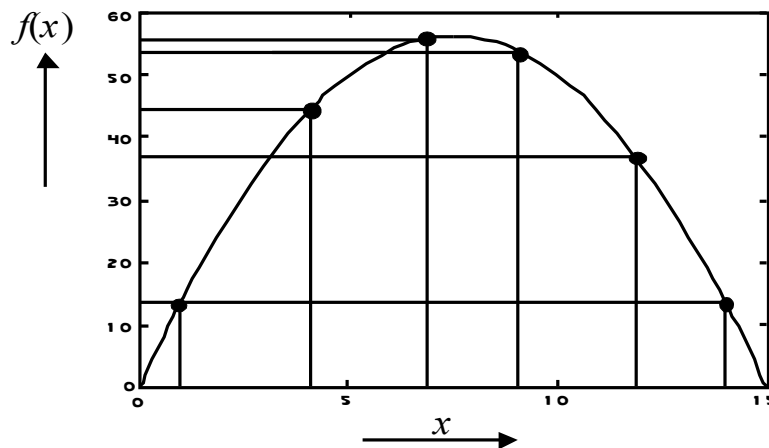
<i>Integer</i>	<i>Binary code</i>	<i>Integer</i>	<i>Binary code</i>	<i>Integer</i>	<i>Binary code</i>
1	0 0 0 1	6	0 1 1 0	11	1 0 1 1
2	0 0 1 0	7	0 1 1 1	12	1 1 0 0
3	0 0 1 1	8	1 0 0 0	13	1 1 0 1
4	0 1 0 0	9	1 0 0 1	14	1 1 1 0
5	0 1 0 1	10	1 0 1 0	15	1 1 1 1

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

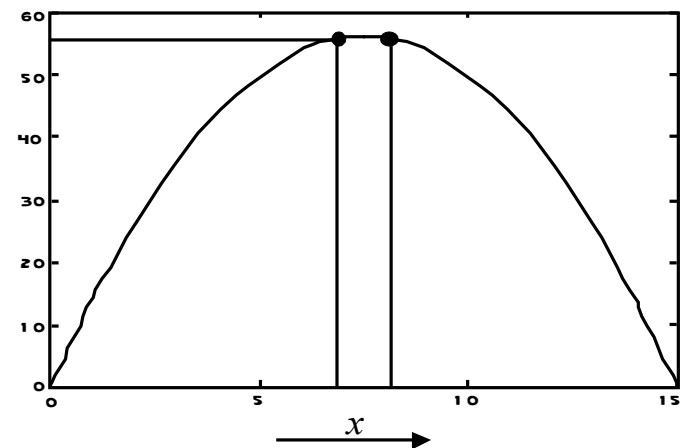
$$f(x) = 15x - x^2$$

# The fitness function and chromosome locations


<i>Chromosome label</i>	<i>Chromosome string</i>	<i>Decoded integer</i>	<i>Chromosome fitness</i>	<i>Fitness ratio, %</i>
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.



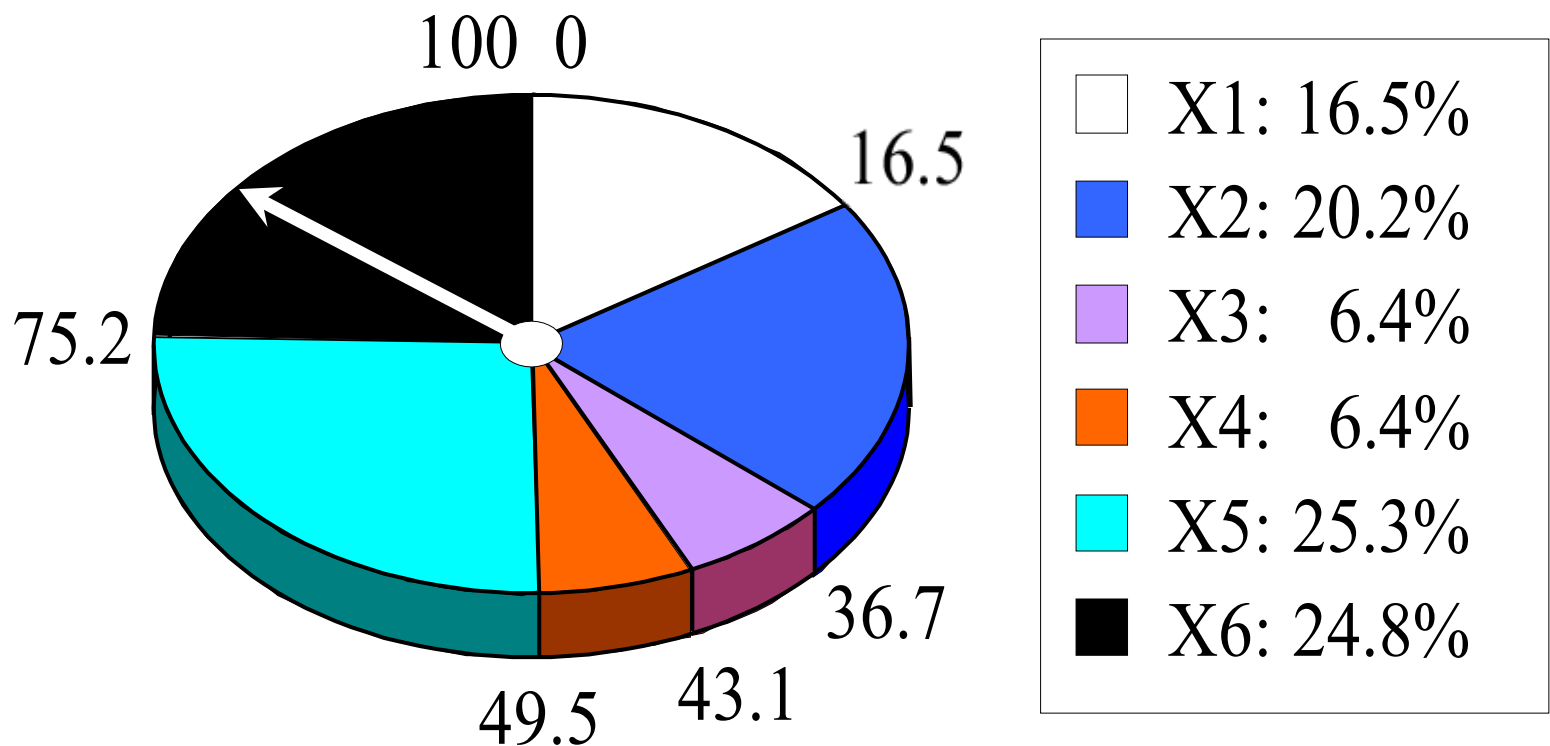
(b) Chromosome final locations.

- 
- The 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

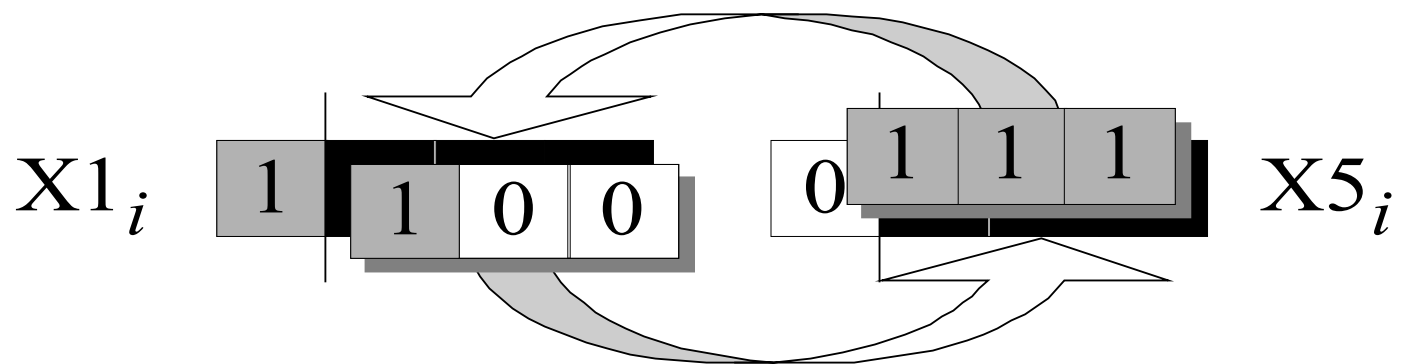
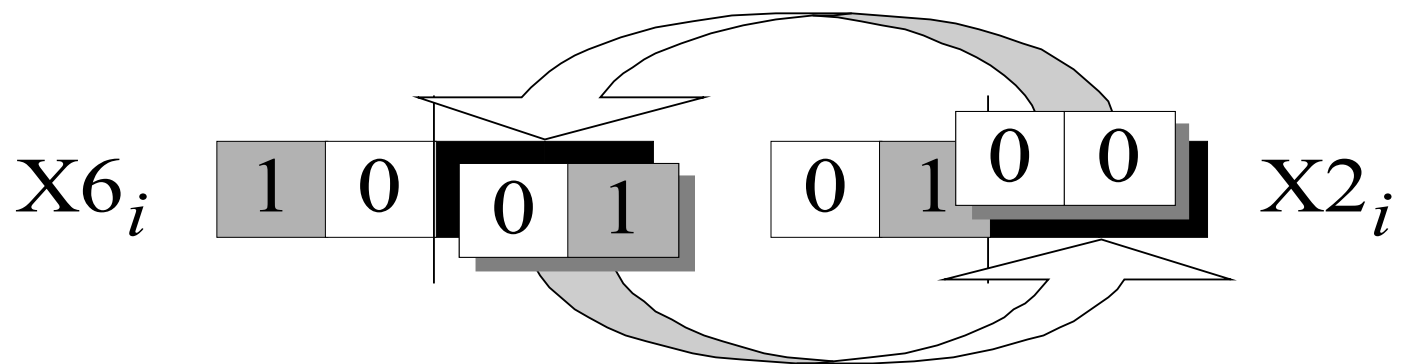
- The most commonly used chromosome selection techniques is the **roulette wheel selection**.





# 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.
- 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.





# Mutation operator

- Mutation represents a change in the gene.
- Mutation is a background operator. Its role is to provide a guarantee that the search algorithm is not trapped on a local optimum.
- The mutation operator flips a randomly selected gene in a chromosome.
- The mutation probability is quite small in nature, and is kept low for GAs, typically in the range between 0.001 and 0.01.

$X6'_i$ 

1	0	0	0
---	---	---	---

$X2'_i$ 

0	1	0	1
---	---	---	---

$X1'_i$ 

1	1	1	1
---	---	---	---

$X5'_i$ 

0	1	0	0
---	---	---	---

$X2_i$ 

0	1	0	0
---	---	---	---

$X5_i$ 

0	1	1	1
---	---	---	---

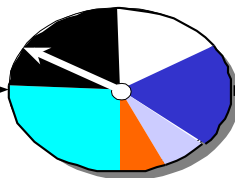
1	0	1	1
---	---	---	---

 $X1''_i$

0	1	1	0
---	---	---	---

 $X2''_i$





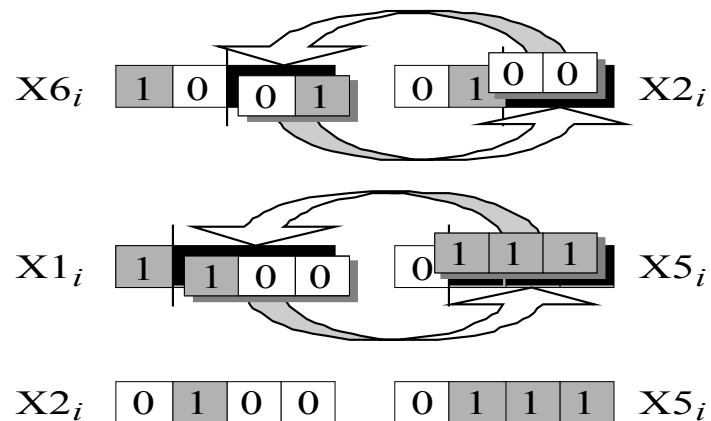
### Generation $i$

$X1_i$	1	1	0	0	$f=36$
$X2_i$	0	1	0	0	$f=44$
$X3_i$	0	0	0	1	$f=14$
$X4_i$	1	1	1	0	$f=14$
$X5_i$	0	1	1	1	$f=56$
$X6_i$	1	0	0	1	$f=54$

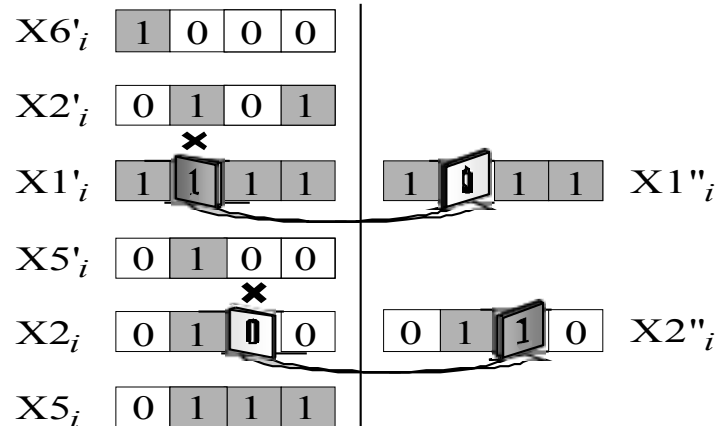
### Generation $(i+1)$

$X1_{i+1}$	1	0	0	0	$f=56$
$X2_{i+1}$	0	1	0	1	$f=50$
$X3_{i+1}$	1	0	1	1	$f=44$
$X4_{i+1}$	0	1	0	0	$f=44$
$X5_{i+1}$	0	1	1	0	$f=54$
$X6_{i+1}$	0	1	1	1	$f=56$

### Crossover

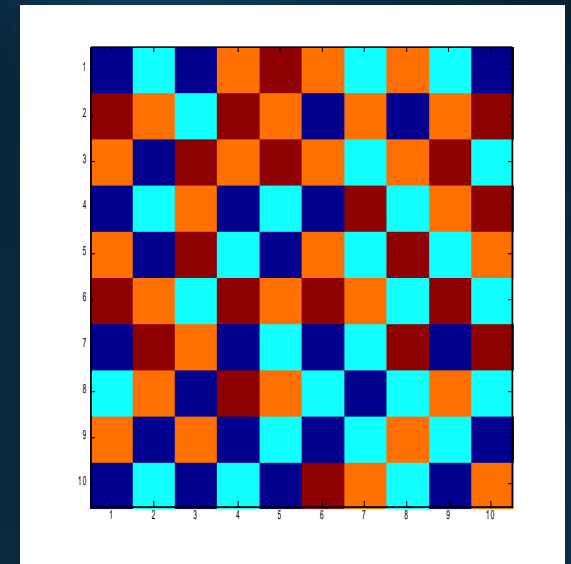
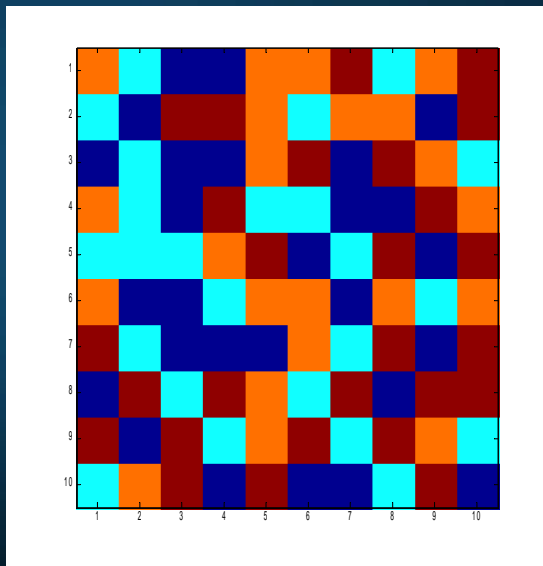


### Mutation



# Example Checkboard

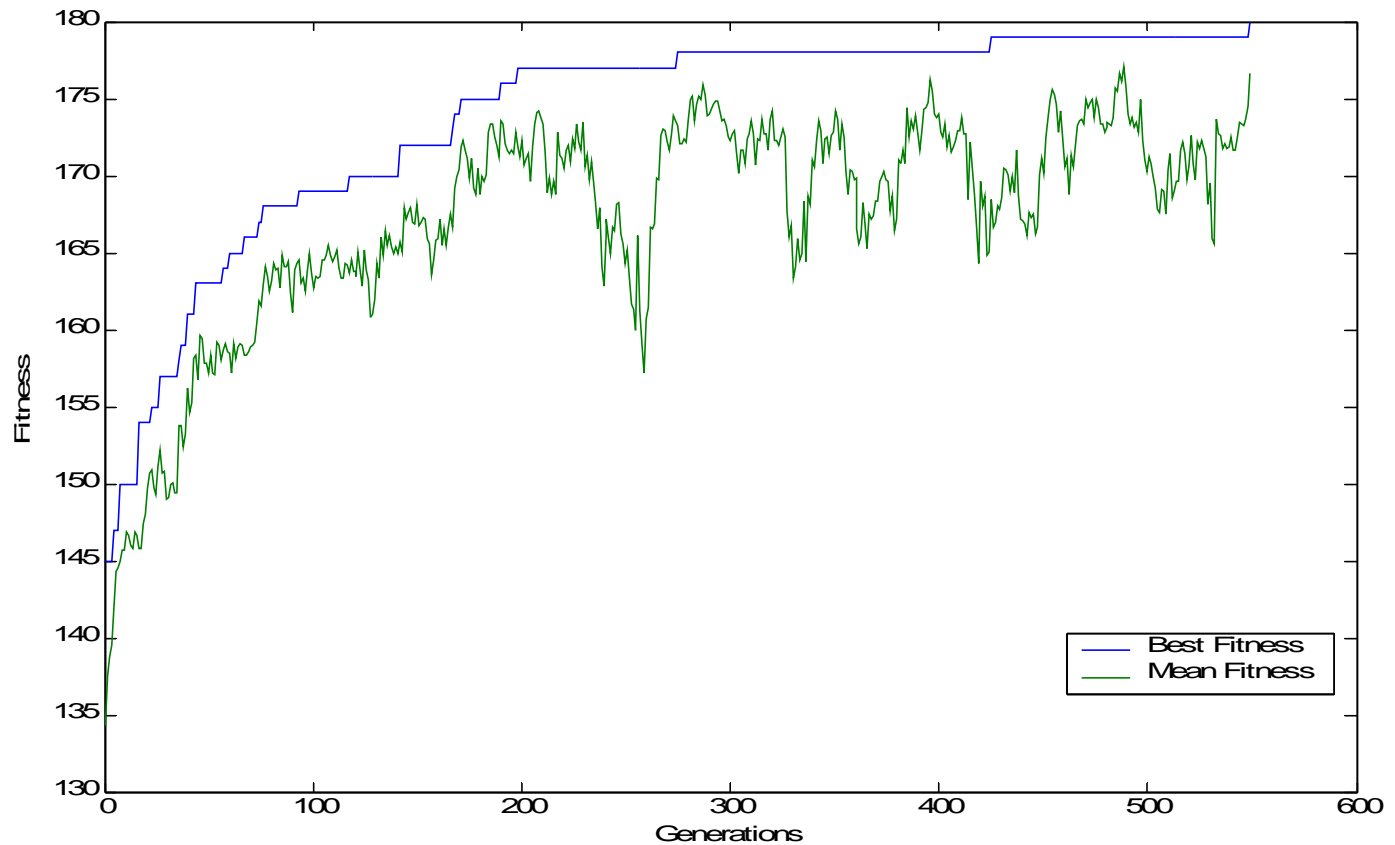
- We are given an  $n$  by  $n$  checkboard in which every field can have a different colour from a set of four colours.
- Goal is to achieve a checkboard in a way that there are no neighbours with the same colour (not diagonal).





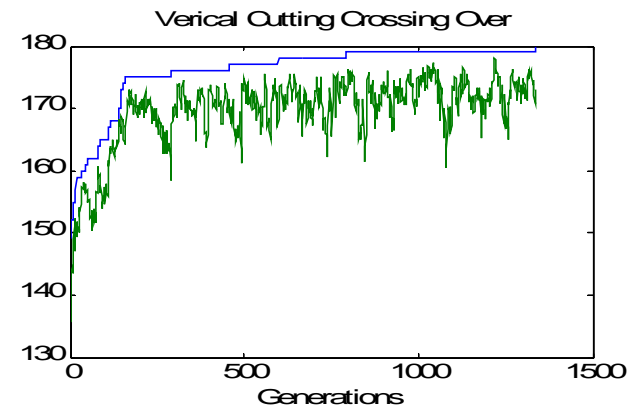
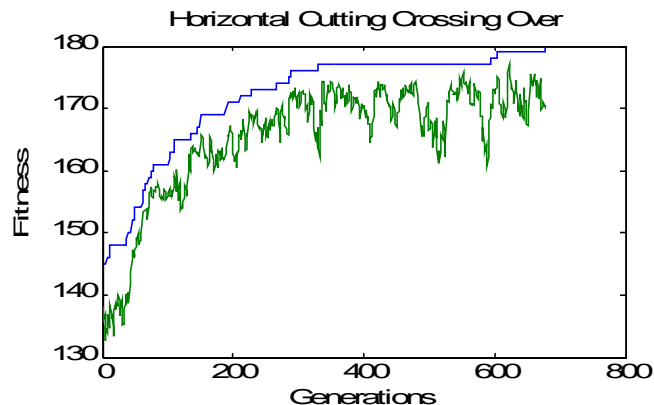
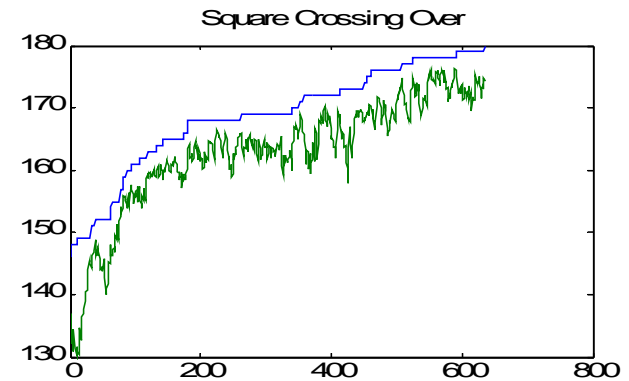
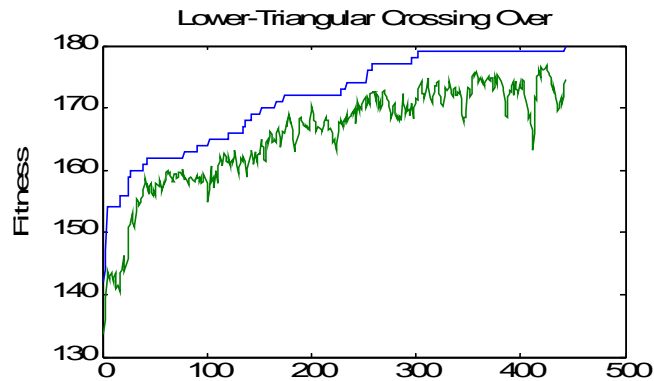
- Chromosomes represent the way the checkboard is coloured.
- Chromosomes are not represented by bitstrings but by bitmatrices
- The bits in the bitmatrix can have one of the four values 0, 1, 2 or 3, depending on the colour
- Crossing-over involves matrix manipulation instead of point wise operating. Crossing-over can be combining the parental matrices in a horizontal, vertical, triangular or square way
- Mutation remains bitwise changing bits in either one of the other numbers

- Fitness curve for the checkboard example



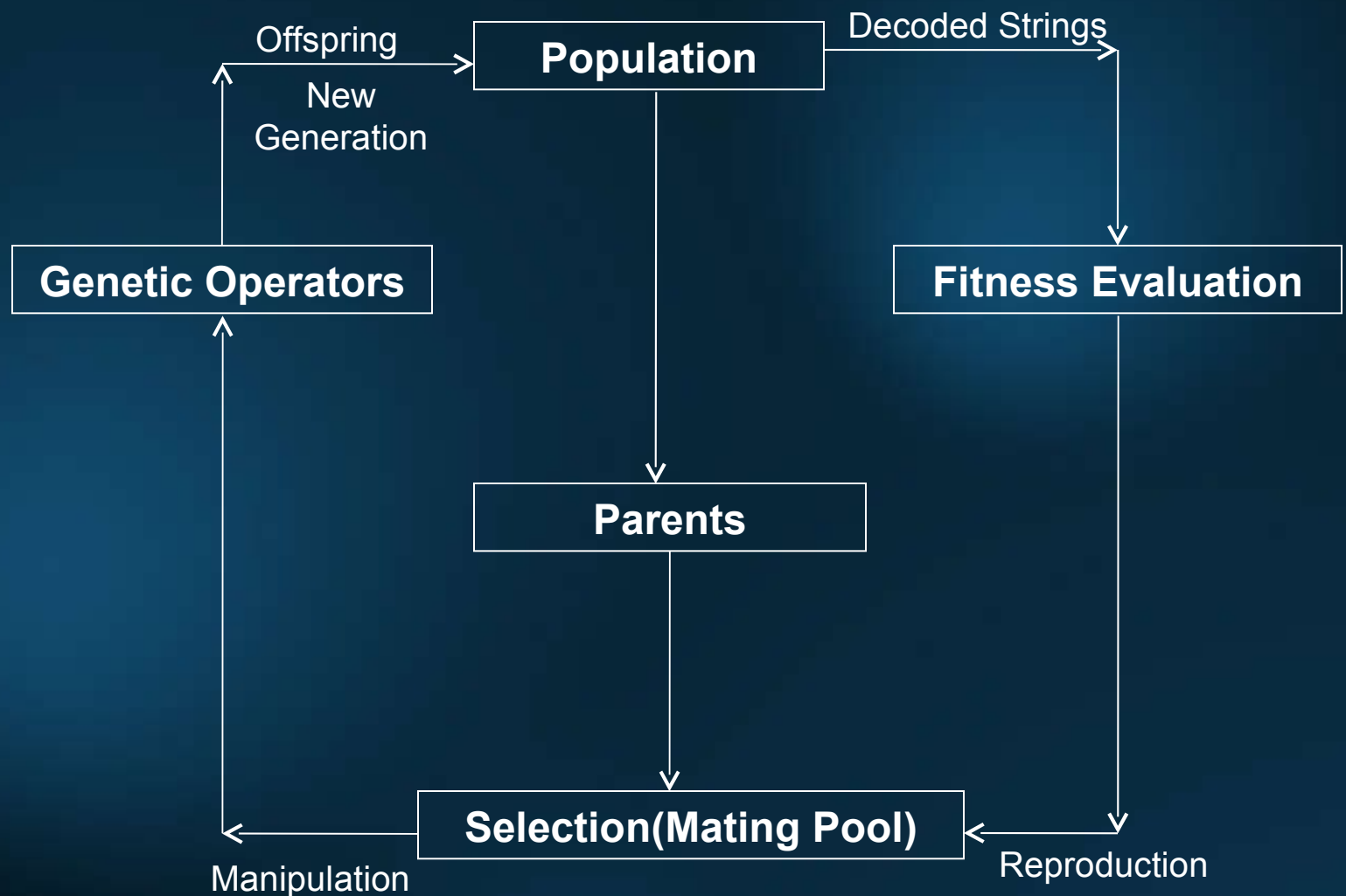
- This problem can be seen as a graph with  $n$  nodes and  $(n-1)$  edges, so the fitness  $f(x)$  is easily defined as:  $f(x) = 2 \cdot (n-1) \cdot n$

# Fitnesscurves for different cross-over rules





# The GA Cycle of Reproduction





# Issues for Genetic Algorithm practitioners

- Choosing basic implementation issues such as
  - Representation
  - Population size and mutation rate
  - Selection, deletion policies
  - Cross over and mutation operators
- Termination criterion
- Performance and scalability
- Solution is only as good as evaluation functions.



# Benefits Of Genetic Algorithms

- Easy to understand
- Supports multi-objective optimisation
- Good for noisy environment
- We always get answer and answer gets better with time
- Inherently parallel and easily distributed
- Easy to exploit for precious or alternate solutions
- Flexible in forming building blocks for hybrid applications
- Has substantial history and range of use



# Genetic Algorithm Applications

Domains	Application Types
Control	Gas pipeline, pole balancing, missile evasion, pursuit
Robotics	Trajectory planning
Signal Processing	Filter design
Game Playing	Poker, checker, prisoner's dilemma
Scheduling	Manufacturing facility, scheduling, resource allocation
Design	Semiconductor layout, aircraft design, keyboard configuration, communication networks
Combinatorial Optimisation	Set covering, travelling salesmen, routing, bin packing, graph coloring and partitioning



# Any Questions







# Bibliography

- [www.softcomputing.net/gpsystems.pdf](http://www.softcomputing.net/gpsystems.pdf)
- <http://cs.mwsu.edu/~simpson/>
- <http://geneticalgorithms.ai-depot.com/Tutorial/Overview.html>
- Neural Networks, Fuzzy Logic, And Genetic Algorithm: Synthesis And Application
  - S.Rajasekaran
  - G.A.Vijayalakshmi Pai
- <http://c2.com/cgi/wiki?GeneticAlgorithm>



Thank You