Date:

Page:

GA - Introduction

* Definition:

→ Genetic Algorithm (GA) is a Search-based optimization technique based on the principles of Genetics & Natural selection.

→ Genetic Algorithm is not Gene coding rather they are computer algorithm that resides or based on principles of genetics & Evolution.

→ It is frequently used to find optimal or near ~~optimal~~ optimal solutions to difficult problems which otherwise would take a lifetime to solve.

→ It is frequently used to solve optimization problems, in machine learning.

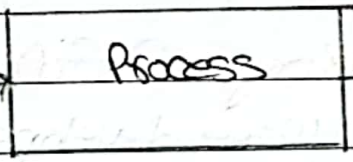
→ GAs are a subset of a much larger branch of computation known as Evolutionary computation.

→ Focus on: Optimization & Searching.

* Optimization:

→ Optimization is the process of making something better.

e.g: In any process, we have a set of inputs & a set of outputs as shown in the figure.



optimization refers to finding the values of inputs in such a way that we get the "best" output values.

Date:

Page:

→ The definitions of "best" varies from problem to problem, but in mathematical terms, it refers to maximizing or minimizing one or more objective functions, by varying the input parameters.

→ The set of all possible solutions or values which the inputs can take make up the search space. In this search space, lies a point or a set of points which gives the optimal solution. The aim of optimization is to find that point or set of points in the search space.

* Searching:

→ The process of finding a solution of the given problem can be defined as searching.

for e.g:- Starting point is given, we need to find the target point.

* Concept of Survival of the fittest:

→ The main principle of evolution used in GA is "Survival of the fittest"

→ The good solution survive, while bad one die.

* The History of GA:

GAs were developed by John Holland and his students & colleagues at the University of Michigan, in 1975.

Date: .....

Page: .....

→ Until ~~80's~~ early 80's, the concept was studied theoretically.

→ In 80s, the first "real world" GAs were designed. Since then GAs are tried on various optimization problems with a high degree of success.

* How GAs works?

In GAs, we have a pool or a population of possible solutions to the given problem. These solutions then undergo crossover & mutation (like in natural genetics) producing new children, 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. This is in line with the Darwinian Theory of "Survival of the fittest".

In this way we keep "evolving" better individuals or solutions over generations, till we reach a stopping criteria.

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).

Date: _____

Page: _____

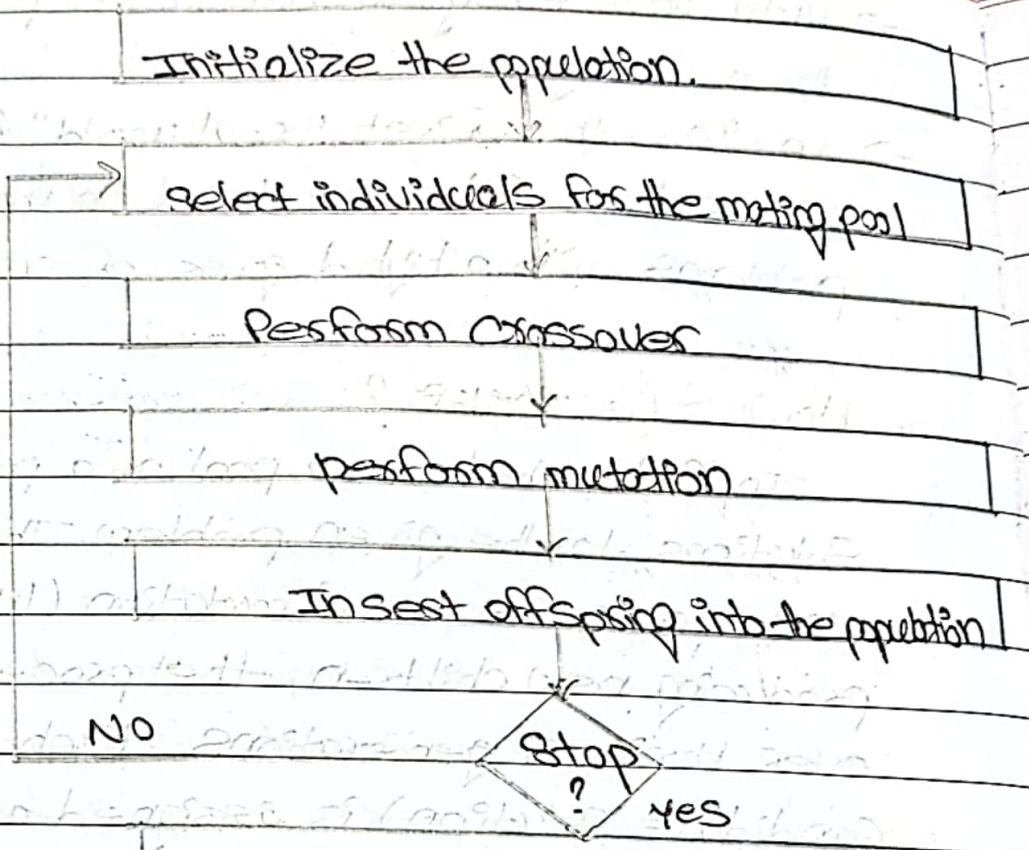


Fig of:- Algorithmic phases of GA.

* Advantages of GA:

GAs have various advantages which have made them immensely popular. These includes:

i) Doesnot require any prior or derivative information (which may not be available for many real-world problems).

ii) It is fastes & more efficient as compared to the traditional methods.

iii) It has very good parallel capabilities,

iv) It can optimize both continous & discrete functions

Date:

Page:

and also multi-objective problems.

v) It provides a list of "good solutions", not just a single solution.

vi) It always gets an answer to the problem, which gets better over the time.

vii) Mostly useful when the search space is very large & there are a large number of parameters involved.

* Limitations of GAs:

i) GAs are not suited for all problems, especially problems which are simple & for which derivative information is available.

ii) Fitness value is calculated repeatedly which might be computationally expensive for some problems.

iii) Being stochastic, there are no guarantees on the optimality or the quality of the solution.

iv) If not implemented properly, the GA may not converge to the optimal solution.

* GA-Motivation:

→ Genetic Algorithms have the ability to deliver a "good-enough" solution "fast-enough". Which makes genetic algorithms attractive for use in solving optimization problems.

Some of the reasons why GAs are needed are as follow:

Date:

Page:

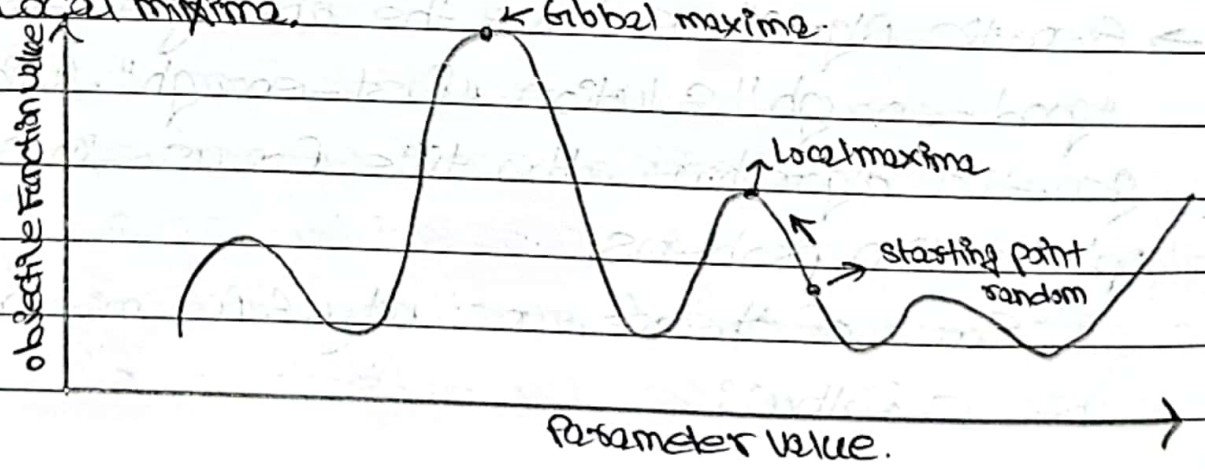
i) Solving difficult Problems:

In Computer Science, there is a large set of problems, which are NP-Hard, which means, even with the most powerful computing systems it take a very long time (even years) to solve that problem. In such scenario, GAs prove to be an efficient tool to provide usable near-optimal solutions in a short amount of time.

ii) Failure of Gradient Based Methods: (Hill climbing problem)

Traditional calculus based methods work by starting at a random point and by moving in the direction of the gradient, till we reach the top of the hill. This technique is efficient & works very well for single-peaked objective functions like the cost function in linear regression.

But in most real-world situations, we have a very complex problem called as landscapes, which are made of many peaks & many valleys, which causes such methods to fails as they get stuck at the local maxima.



Date: .....

Page: .....

iii) Getting a Good Solution Fast :

Some difficult problems like the Travelling Salesperson problem (TSP), have real-world applications like path finding & VLSI (Very large Scale Integration) Design. Mostly used in GPS system for finding the quickest & shortest path from source to destination.

* Basic Terminology:

Before beginning a discussion on Genetic Algorithm it is essential to be familiar with some basic terminology which will be used throughout this tutorial.

i) Population:

It is a subset of all the possible (encoded) solutions to the given problem. The population for a GA is similar to the population for human beings except that instead of human beings, we have candidate solutions representing human beings.

ii) Chromosome:

A chromosome is one such solution to the given problem.

iii) Gene:

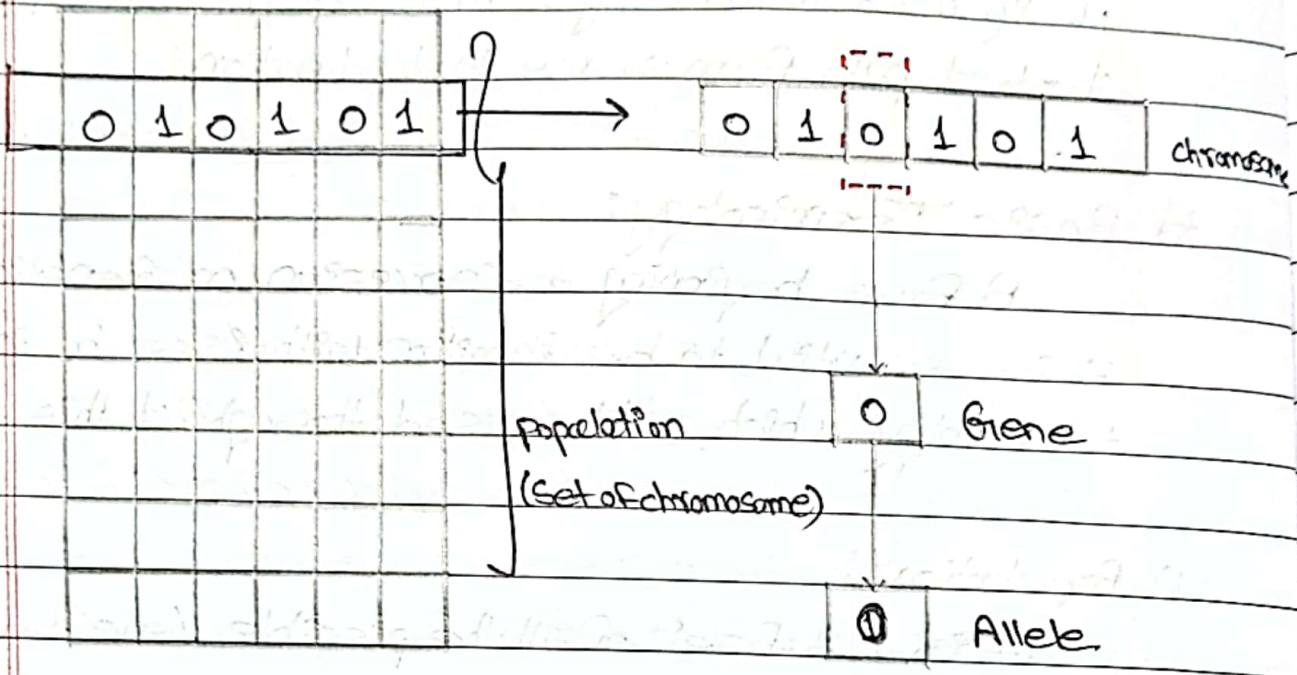
A gene is one element position of a chromosome.

Date: .....

Page: .....

iv) Allele:

It is the value a gene takes for a particular chromosome.



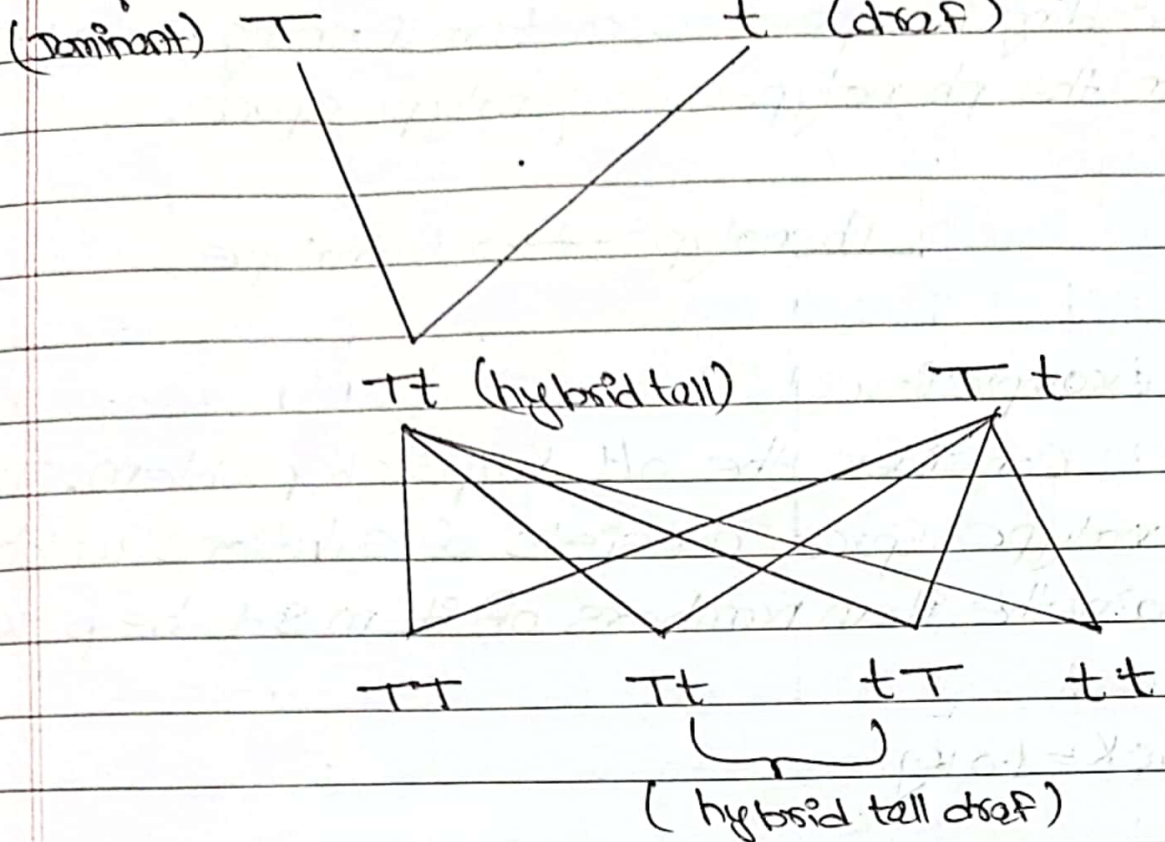
v) Genotype:

Genotype is the population in the Computational Space. In the Computational Space, the solutions are represented in a way which can be easily understood & manipulated using a computing system.

vi) Phenotype:

Phenotype is the population in the actual real world situation / solution space in which solutions are represented in a way that are represented in real world situations.

eg:-



Genotype: 1 2 1

phenotype: 3 : 1

* Encoding & Decoding:

For simple problems, the phenotype & Genotype spaces are the same. However, in most of the cases, the phenotype & genotype spaces are different.

;) Decoding:

→ Decoding is a process of transforming a solution from the genotype to the phenotype space.

→ Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation

Genotype → phenotype

Date:

Page:

ii) Encoding:

→ Encoding is a process of transforming the solution from the phenotype to genotype space.

Phenotype → Genotype

* For example:-

Consider the 0/1 Knapsack problem. The phenotype space consists of solutions which just contain the item numbers of item s to be picked.

i.e.

Sack = 10 kg

Kg	5	3	6	4	2	} phenotype space solution
Values	6	7	12	7	5	

However, the genotype space it can be represented as a binary string of length n (where n is the number of items).

A 1 at position x represents that x^{th} item is picked while a 0 represents the reverse.

This is a case where genotype & phenotype spaces are different.

Solution 1: | 1 | 1 | 0 | 0 | 1 |

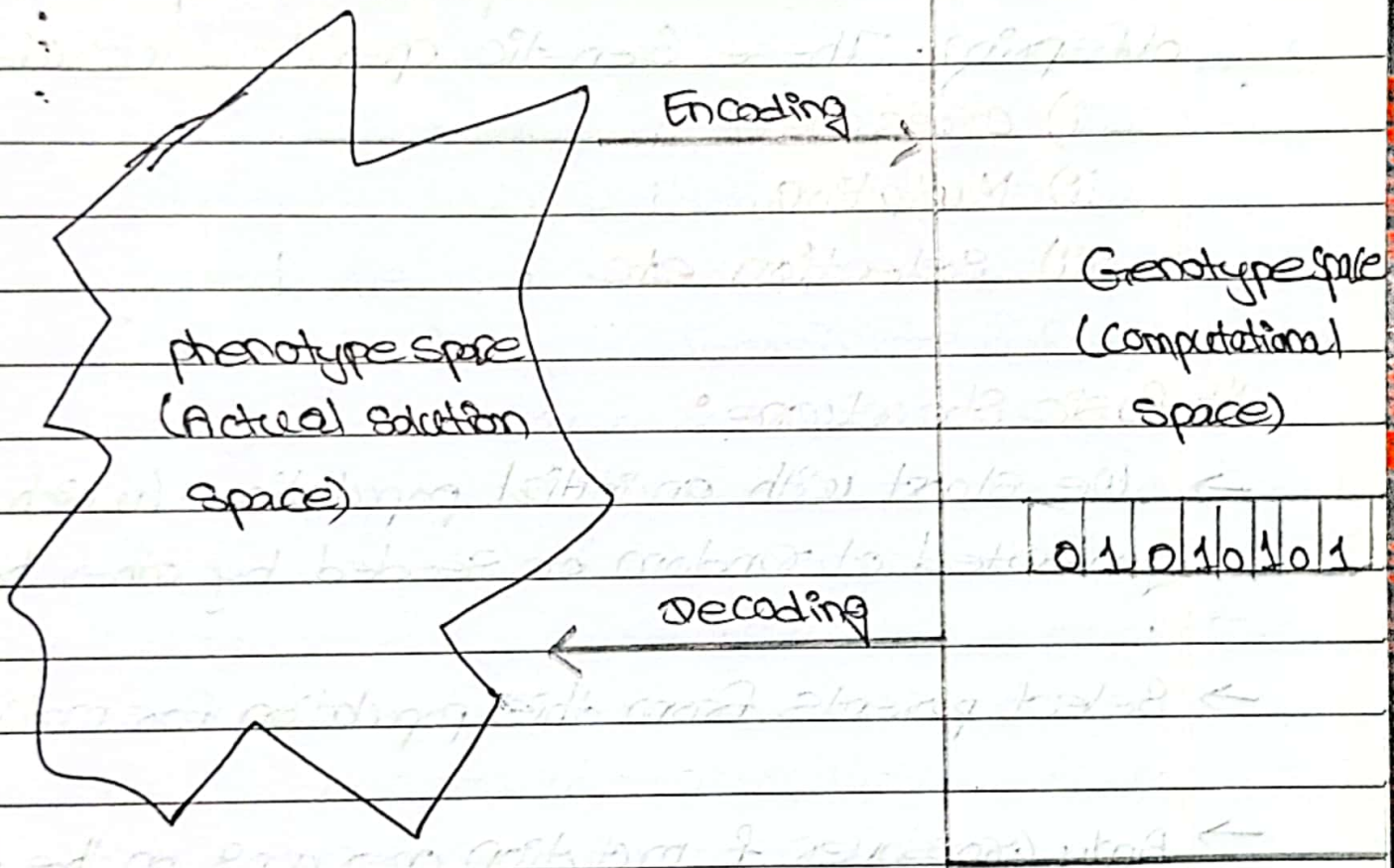
Solution 2: | 1 | 0 | 0 | 1 | 0 |

Solution 3: | 0 | 1 | 1 | 0 | 0 |

Solution 4: | 0 | 0 | 1 | 1 | 0 |

Genotype Space Solution.

Solution 5: | 1 | 1 | 1 | 0 | 0 | X exceeds the 10kg



Date: _____

Page: _____

vii) Fitness Function:

A fitness function 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 problems.

viii) Genetic operators:

These alter the genetic composition of the offspring. These Genetic operators include:

i) Crossover,
ii) Mutation,
iii) Selection etc.

* Basic Structure:

→ We start with an initial population (which may be generated at random or seeded by other heuristics)

→ Select parents from this population for mating.

→ Apply crossover & mutation operators on the parents to generate new off-springs.

→ Finally, these off-springs replace the existing.

Date: .....

Page: .....

individuals in the population and the process repeats.

In this way genetic algorithms actually try to mimic the human evolution to some extent.

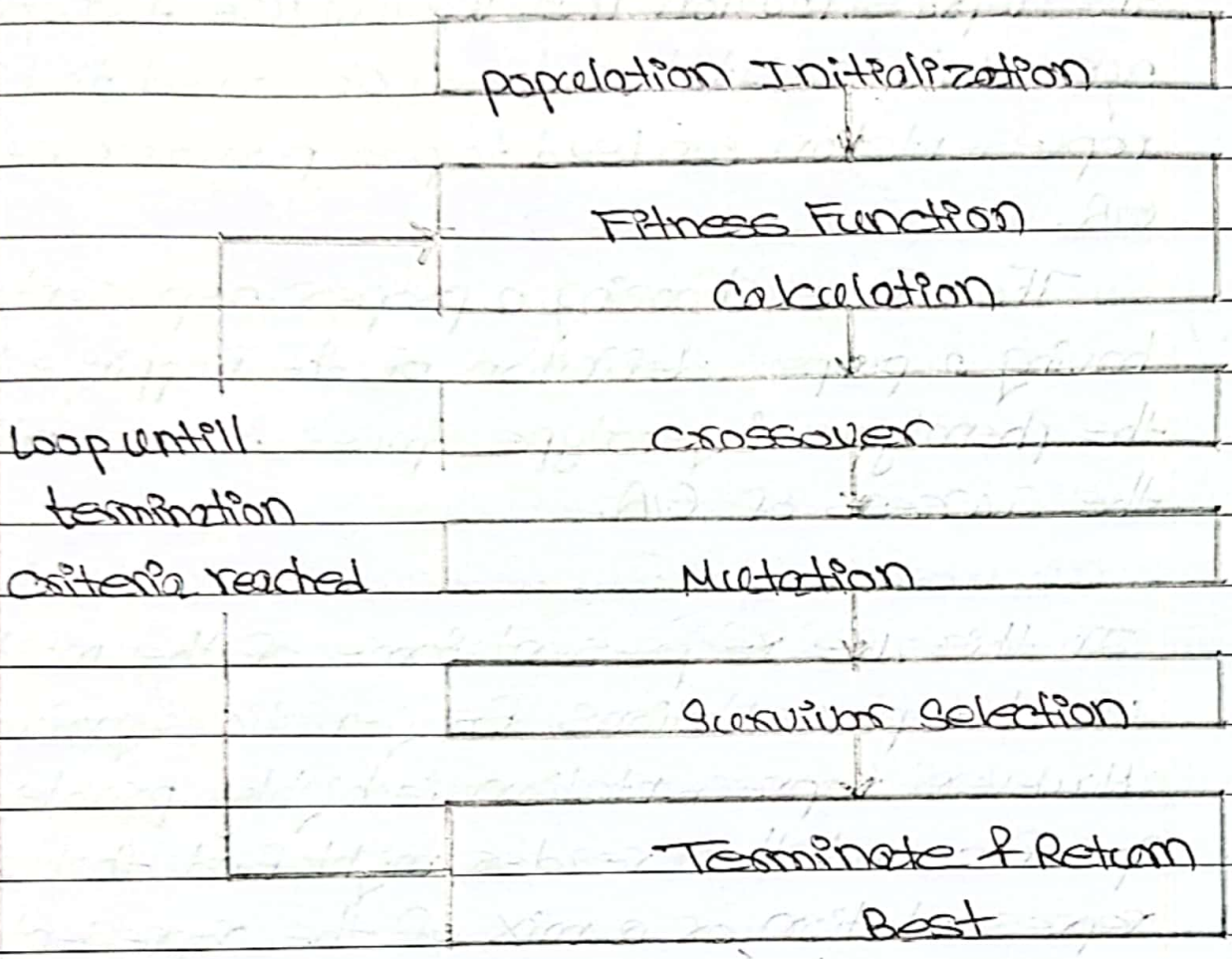


Fig of : Basic Structure of G.A.

Date: _____

Page: _____

* Genotype Representation:

One of the most important decisions to make while implementing a genetic algorithm is deciding the representation that we will use to represent our solutions. It has been observed that improper representation can lead to poor performance of the GA.

Therefore, choosing a proper representation, having a proper definition of the mappings between the phenotype & genotype spaces is essential for the success of GA.

In this, we present some of the most commonly used representations for genetic algorithms. However, representation is highly problem specific and the reader might find that another representation or a mix of the representations mentioned here might suit their problem better.

i) Binary Representation:

This is one of the simplest & most widely used representation in GAs. In this type of representation the genotype consists of bit strings.

For some problems when the solutions space consists of Boolean decision variables — yes or no, the binary representation is best suited.

Date: .....

Page: .....

Take for example the 0/1 knapsack problem, If there are n items, we can represent a solution by a binary string of n elements, where the X^{th} element tells whether the item X is picked (1) or not (0).

0	0	1	0	1	1	1	0	0	1

For other problems, specifically those dealing with numbers, we can represent the numbers with their binary representation. The problem with this kind of encoding is that different bits have different significance and therefore mutation & crossover operators can have undesired consequences. This can be resolved to some extent by using Gray Coding, as a change in one bit does not have a massive effect on the solution.

ii) Real Valued Representation:

For Problems where we want to define the genes using continuous rather than discrete variables, the real valued representation is the best suited. The precision of these real valued or floating point numbers is however limited to the computer.

0.5	0.2	0.6	0.8	0.7	0.4	0.3	02	0.1	09

Date: _____

Page: _____

iii) Integer Representation:

For discrete valued genes, we cannot always limit the solution space to binary 'yes' or 'no'. For example, if we want to encode the four directions {East, West, North, South}, we can encode them as {0, 1, 2, 3}. In such cases, integer representation is ~~desirible~~ best suited.

	1	2	3	4	3	2	4	1	2	1

iv) Permutation Representation:

In many problems, the solution is represented by an order of elements. In such cases permutation representation is the most suited.

A classical 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 main aim of this problem is 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.

Date:

Page:

1	5	9	8	7	4	2	3	6	0

* Population:

Population is a subset of solutions in the current generation. It can also be defined as a set of chromosomes. There are several things to be kept in mind when dealing with GA population. i.e:-

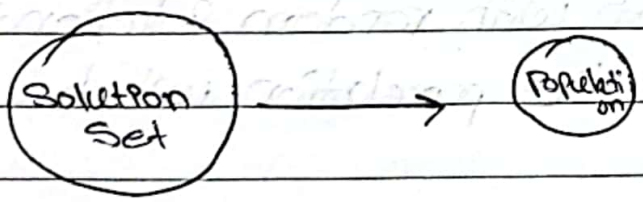
i) The diversity of the population should be maintained otherwise it might lead to premature convergence.

ii) The population size should not kept very large as it can cause a GA to slow down, while a smaller population might not be enough for a good mating pool.

Therefore, an optimal population size needs to be decided by trial and error.

The population is usually defined as a two dimensional array of :- i) size population, size X

ii) Chromosome size.



Date: .....

Page: .....

* Population Initialization:

There are two primary methods to initialize a population in a GA. They are:

i) Random Initialization:

Populate the initial population with completely random solutions.

ii) Heuristic Initialization:

Populate the initial population using a known heuristic for the problem.

→ It has been observed that the entire population should not be initialized using a heuristic, as it can result in the population having similar solutions and very little diversity.

→ It has been experimentally observed that the random solutions are the ones to drive the population to optimality.

→ Therefore, with heuristic initialization, we just seed the population with a couple of good solutions, filling up the rest with random solutions rather than filling the entire population with heuristic based solutions.

Date: .....

Page: .....

It has also been observed that heuristic initialization in some cases, only effects the initial fitness of the population, but in the end, it is the diversity of the solutions which lead to optimality.

* Population Models:

There are two population models widely in use:

i) Steady State:

In steady state GA, we generate one or two off-springs in each iteration & they replace one or two individuals from the population. A steady state GA is also known as Incremental GA.

ii) Generational:

In a generational model, we generate 'n' off-springs, where n is the population size, and the entire population is replaced by the new one at the end of the iteration.

Date: .....

Page: .....

Fitness Function:

A fitness function simply defined is a functi which takes a candidate solution to the proble as input & produces an output and also determine how "fit" or how "good" the solution is with respect to the problem in consideration.

The calculation of fitness value is done repeatedly in a GA and therefore it should be sufficiently fast. A slow computation of the fitness value can adversely affect a GA and make it exceptionally slow.

In most cases the fitness function and the objective function are the same as the objectiv is to either maximize or minimize the given objective function. However, for more complex problems with with multiple objectives & constraints, an Algorithm Designer might choose to have a different fitness function.

V.V.I

When the objective Function map to Fitness Function?

When the objective of the problem is to find optimal or maximize value both objective function & fitness function can be same.

for e.g:- 0/1 knapsack problems.

Both objective & Fitness Function: Maximum or optimal value.

Date: .....

Page: .....

but when the objective is to find the minimize value objective function & fitness function are different as Fitness function always find the optimal solution. So in this case, the objective Function can't be map with Fitness Function.

for eg:- Travelling salesman problem

objective Function : Minimum value

Fitness Function : Maximize value

* Characteristics:

A fitness Function should possess the following characteristics:

i) The fitness function should be sufficiently fast to compute

ii) It should quantitatively measure how fit a given solution is or how fit individuals can be produced from the given solution.

In some cases, calculating the fitness function directly might not be possible due to the inherent complexities of the problem at hand. In such cases we do fitness approximation to suit our needs.

Date: _____

Page: _____

* Example:

The following image shows the fitness calculation for a solution of the 0/1 Knapsack. It is a simple fitness function which just sums the profit values of the items being picked (which have a 1), scanning the elements from left to right till the knapsack is full.

Item Number:	0	1	2	3	4	5	6
chromosome:	0	1	1	1	1	0	0
Profit values:	2	9	8	5	4	0	2
weight values:	7	5	3	1	5	9	8

Knapsack capacity: 15 (14 taken with max profit)

Total resultant profit: 26

* Parent Selection:

Parent selection is the process of selecting parents which mate together to create off-springs for the next generation. Parent selection is very crucial to the convergence rate of the GA as good parents drive individuals to a better & fitter solutions.

However, case should be taken to prevent one extremely fit solution from taking over the entire population in a few generations, as this leads to the solutions being close to one another in the solution space thereby leading to a loss of diversity. This taking up of the entire population by one extremely fit solution is known as Premature Convergence and is an undesirable condition in a GA. Maintaing good diversity in the population is extremely crucial for the success of a GA.

1) * Fitness Proportionate Selection:

Fitness Proportionate Selection is one of the most popular ways of parent selection. In this every individual can become a parent with a probability which is proportional to its fitness.

Therefore, fitter individuals have a higher chance of mating & propagating their features to the next generation. This kind of selection strategy

Date: _____

Page: _____

applies the selection of the more fit individuals from the population, evolving better individuals over time.

Consider a circular wheel. The wheel is divided into n pieces, where n is the number of individuals in the population. Each individuals gets a portion of the circle which is proportional to their fitness values.

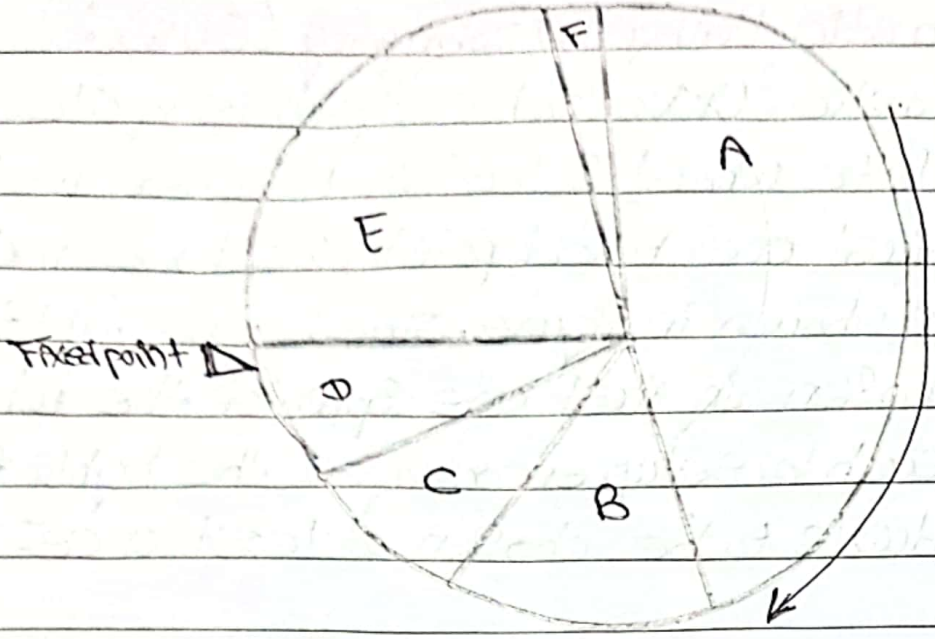
Two implementations of fitness proportionate selection are: i) Roulette wheel selection,
ii) Stochastic Universal Sampling (SUS),

i) Roulette wheel selection:

In roulette wheel selection, the circular wheel is divided into slots of region as described earlier. A fixed point is chosen on the wheel circumference as shown and wheel is rotated. The region of the wheel which comes in front of the fixed point is chosen as the parent. For the second part, the same process is repeated.

Date: .....

Page: .....



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

choose D as the parent.

It is clear that a fitter individual has a greater portion on the wheel and therefore a greater chance of landing in front of the fixed point when the wheel is rotated.

Therefore, the probability of choosing an individual depends directly on its fitness.

*Implementation:

We use the following steps:

Calculate S = the sum of all fitnesses.

Generate a random number between 0 & S.

Starting from the top of the population, keep adding the fitnesses to the partial sum P, till P < S.

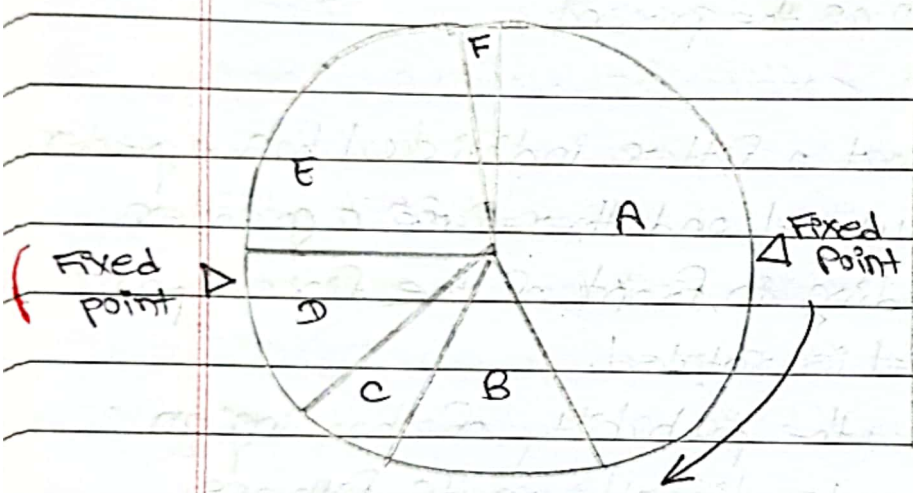
The individual for which P exceeds S is the chosen individual.

Date: _____

Page: _____

ii) Stochastic Universal Sampling (SUS):

Stochastic Universal Sampling is quite similar to Roulette Wheel Selection, however instead of having just one fixed point, we have multiple fixed point as shown in figure. Therefore, all the parents are chosen in just one spin of the wheel. Also, such a setup encourages the highly fit individuals to be chosen at least once.



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

Note: It is to be noted that fitness proportionate selection methods don't work for cases where the fitness can take a negative value.

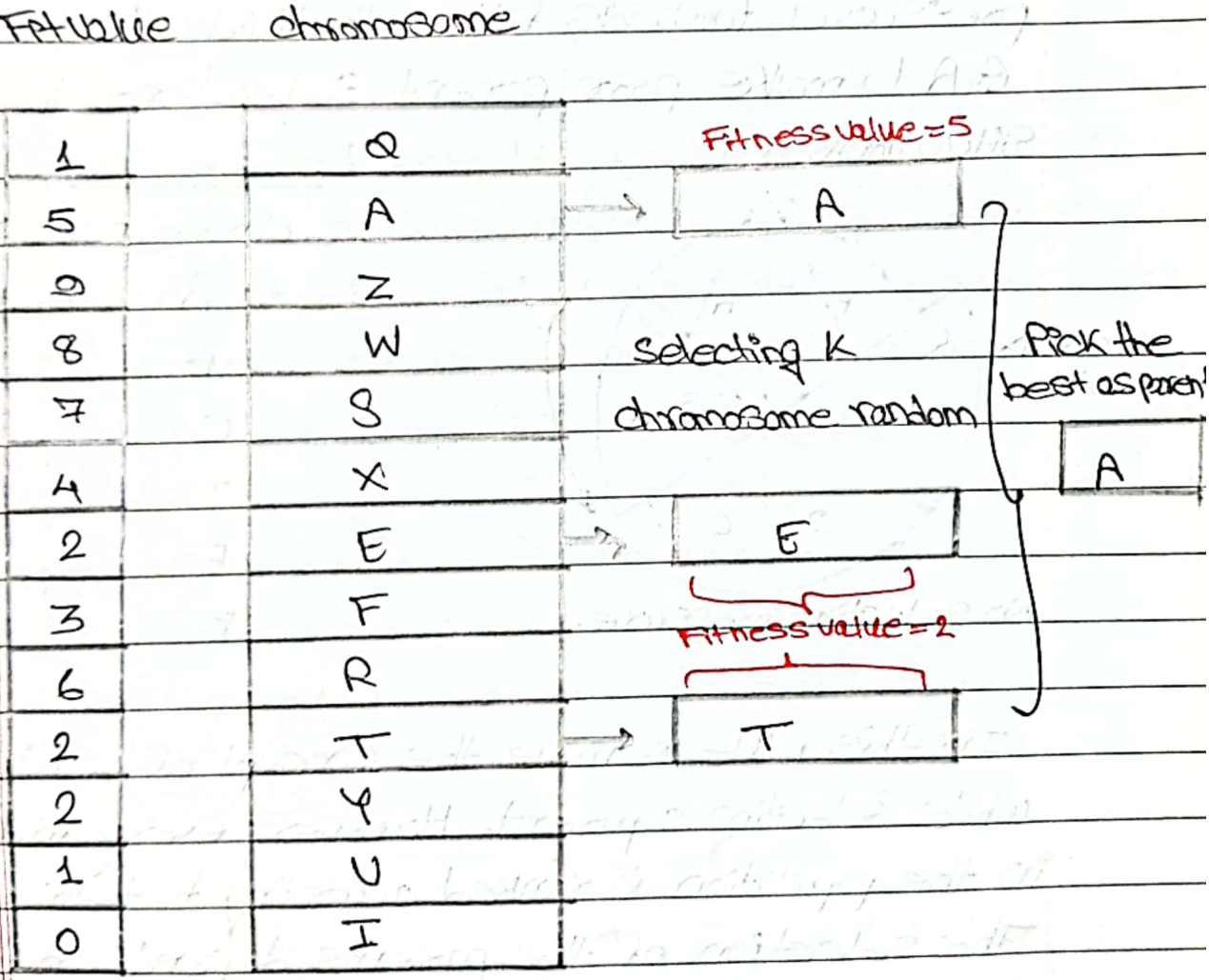
Date:

Page:

2) Tournament Selection:

In K-way tournament selection, we select K individuals from the population at random & select the best out of these to become a parent. The same process is repeated for selecting the next parent.

Tournament selection is also extremely popular in literature as it can even work with negative fitness values.

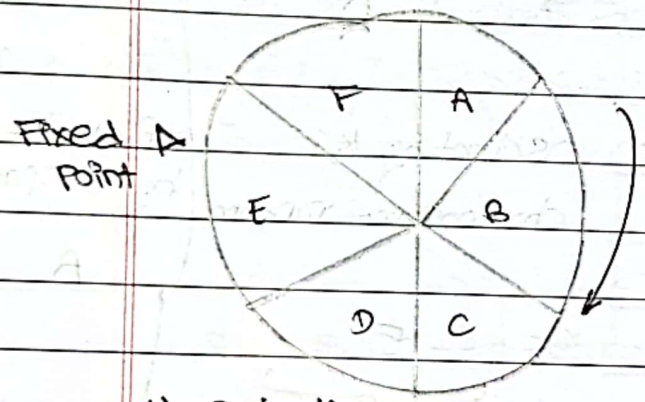


Date: _____

Page: _____

3) Rank Selection:

Rank selection also works with negative fitness values & is mostly used when the individuals in the population have very close fitness values. This leads to each individuals having an almost equal share of the pie (like in case of fitness proportionate selection) as shown in the following figure & hence each individual in the population has same probability of getting selected as a parent. This in turn leads to a loss in the selection pressure towards fitter individuals, making the GA to make poor parent selections in such situations.



No selection pressure.

chromosome	Fitness value
A	8.1
B	8.0
C	8.05
D	7.95
E	8.02
F	7.99

In this, we remove the concept of a fitness value while selecting a parent. However, every individual in the population is ranked according to their fitness. The selection of the parents depends on the rank of each individual & not the fitness. The higher ranked

Date:

Page:

individuals are preferred more than the lower ranked ones.

chromsome	Fitness value	Rank
A	8.1	1
B	8.0	4
C	8.05	2
D	7.95	6
E	8.02	3
F	7.99	5

4) Random Selection:

In this Strategy we randomly select parents from the existing population. There is no selection pressure towards fitter individuals and therefore this strategy is usually avoided.

Date: _____

Page: _____

* Crossover:

* Introduction:

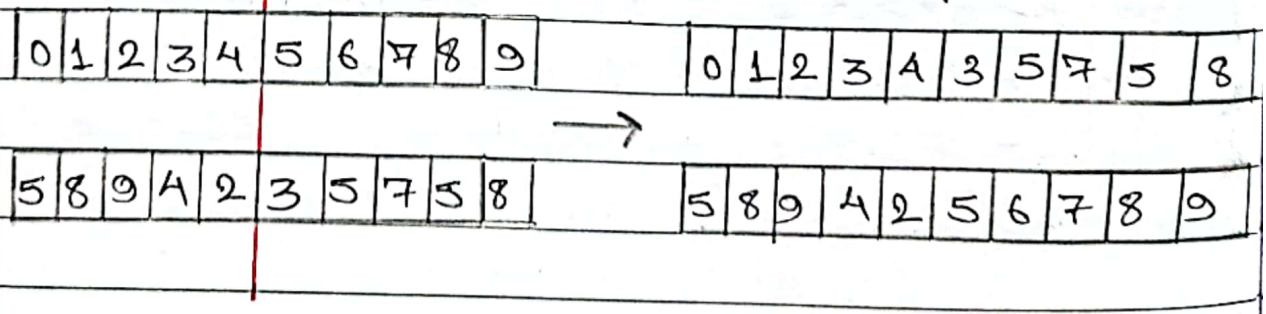
The Crossover operator is similar to reproduction & biological crossover. In this more than one parent is selected and ~~one~~ one or more off-springs are produced using the genetic material of the parents. crossover is usually applied in a GA with a high probability.

* Crossover operators:

In this we will dicuss some of the most popularly used crossover operators. It is to be noted that these crossover operators are very generic & the GA designer might choose to implement a problem-specific crossover operator as well.

i) One-point crossover:

In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs.

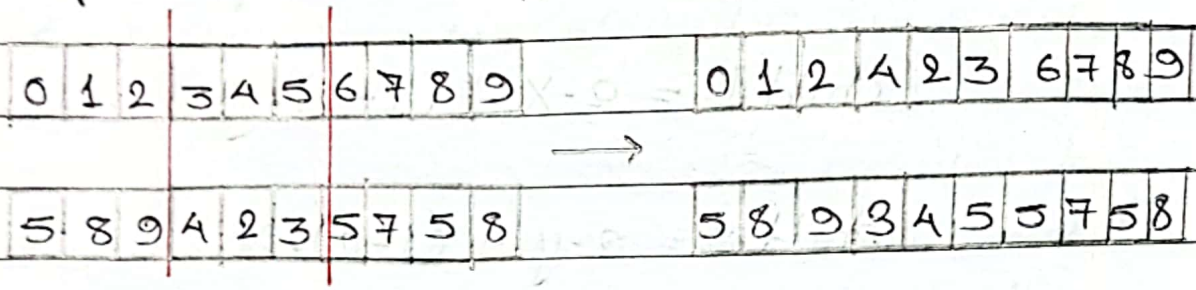


Date: .....

Page: .....

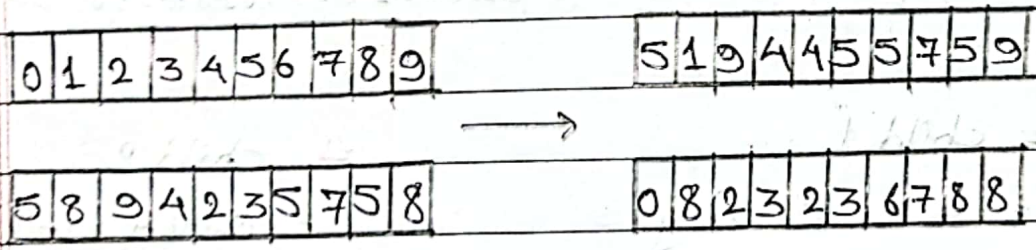
ii) Multi-Point Crossover:

Multipoint crossover is a generalization of the one-point crossover where the alternating segments are swapped to get new off-springs.



iii) Uniform Crossover: (3-point)

In a uniform crossover, we don't divide the chromosome into segments, rather we treat each gene separately. In this, we essentially flip a coin for each gene to decide whether or not it'll be included in the off-spring. We can also bias the coin to one parent, to have more genetic material in the child from that parents.



Date: .....

Page: .....

iv) Whole Arithmetic Recombination:

This is commonly used for integer representations & works by taking the weighted average of the two parents by using the following formula:

$$\text{child 1} = a \cdot x + (1-a) \cdot y$$

$$\text{child 2} = a \cdot y + (1-a) \cdot x$$

where, $a \in [0, 1]$

Also, when $a = 0.5$

Then, both the children will be identical as shown in the following,

0.1	0.1	0.2	0.2	0.3	0.3	0.4	0.4	0.5	0.5
x									

0.2	0.3	0.2	0.2	0.3	0.2	0.3	0.2	0.3	0.2
y									

→

*									
0.15	0.2	0.2	0.2	0.3	0.25	0.35	0.3	0.2	0.35

#									
0.15	0.2	0.2	0.2	0.3	0.25	0.35	0.3	0.2	0.35

* = child 1

$= a \cdot x + (1-a) \cdot y$

$= 0.5 * 0.1 + (1-0.5) * 0.2$

$= 0.05 + 0.1$

$= 0.15$

\# child 2

$= a \cdot y + (1-a) \cdot x$

$= 0.5 * 0.2 + (1-0.5) * 0.1$

$= 0.1 + 0.05$

$= 0.15$

* Similarly For all gene

Date:

Page:

v) Davis' order Crossover (OX1):

i) OX1 (Order crossover):

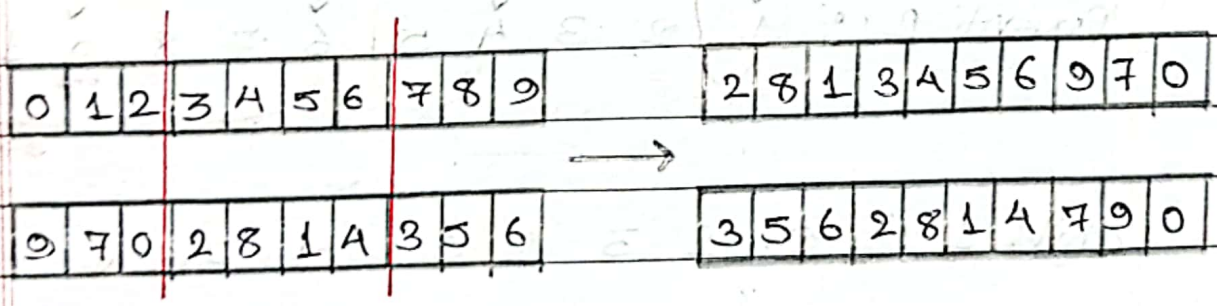
OX1 is used for permutation based crossovers with the intention of transmitting information about relative ordering to the off-springs.

* How it works:

i) Create two random crossover points in the parent & copy the segment between these crossover points from first parent to the first child.

ii) Now, Starting from the second crossover point in the second parent, copy the remaining unused gene numbers from the second parent to the first child, wrapping around the list.

iii) Repeat for the second child with the parent's role reversed.



Date: _____

Page: _____

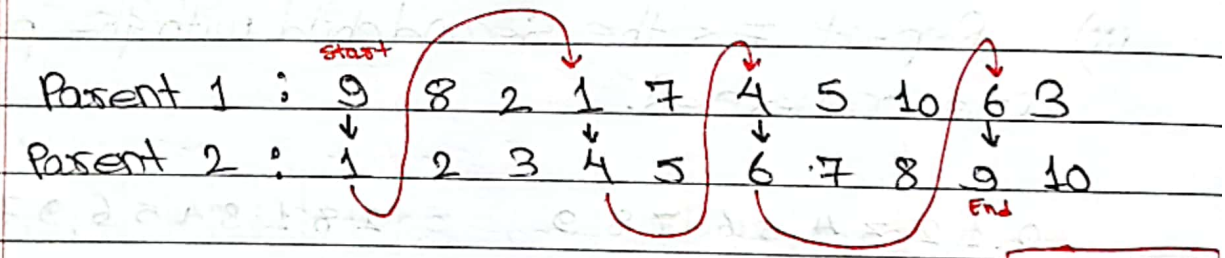
ii) CX (Cyclic crossover):

Cyclic crossover is also a type of Permutation based crossovers. It is specifically applied to problems that involve permutations, such as Travelling Salesman problem (TSP) or job scheduling

* How it works:

i) The cyclic crossoverss starts by identifying cycles within the parent solutions. A cycle is a sequence of elements that cycle through the permutations of the two parents.

ii) The child or offspring are created by swapping the elements between the two parents within the identified cycles.



Child 1 : 9 2 3 1 5 4 7 8 6 10

Child 2 : 1 8 2 4 7 6 5 10 9 3

Date:

Page:

iii) PMX (Partially Matched Crossover):

Partially matched crossover is also one of the type of permutation based crossovers.

① eg:-

Parent 1 : 9 8 4 5 6 7 1 3 2 10

Parent 2 : 8 7 1 2 3 10 9 5 4 6

child 1 : 9 8 4 2 3 10 1 6 5 7

child 2 : 8 10 1 5 6 7 9 2 4 3

② Another eg:- when the segment is large:

Parent 1 : 9 8 4 5 6 7 1 3 2 10

Parent 2 : 8 7 1 2 3 10 9 5 4 6

child 1 : 1 8 6 2 3 10 9 5 4 7

child 2 : 8 10 9 5 6 7 1 3 2 4

③ Parent 1 : A B | C D E F | G

Parent 2 : E D | F G B A | C

child 1 : C E F G B A D

child 2 : B G C D E F A

Date:

Page:

other crossover also exists like shuffle crossover, Ring crossover etc.

* Mutation:

In simple terms, Mutations may be defined as a small random twist in the chromosome, to get a new solution. It is used to maintain and introduce diversity in the genetic population & is usually applied with a low probability. If the mutation probability is very high, the GA gets reduced to a random search.

Mutation is also the part of the GA which is related to the "exploration" of the search space. It has been observed that mutation is essential to the convergence of the GA while crossover is not.

* Mutation Operators:

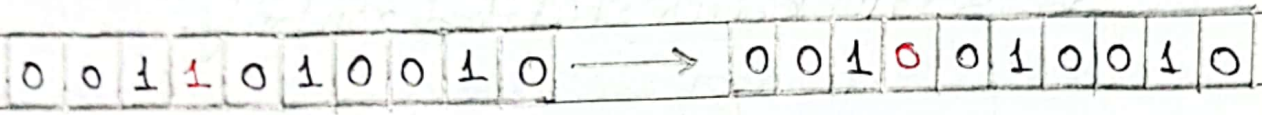
We will discuss some of the most commonly used mutation operators. Like the crossover operators, this is not an exhaustive list & the GA designer might find a combination of these approaches or a problem-specific mutation operator more useful.

Date:

Page:

i) Bit-Flip Mutation:

In this bit flip mutation, we select one or more random bits & flip them. This is used for binary encoded GAs.



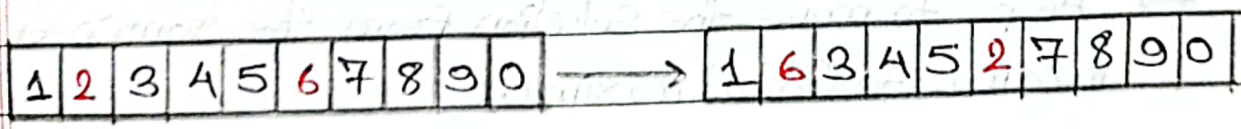
ii) Random Resetting:

Random Resetting is an extension of the bit flip for the integer representation. In this, a random value from the set of allowed values is assigned to a randomly choosen gene.

i.e. $1 \rightarrow 0$
$0 \rightarrow 0$
$8 \rightarrow 0$
} Reset to 0 where $0 \in$ allowed values set

iii) Swap Mutation:

In swap mutation, we select two position on the chromosome at random, and interchange their values. This is commonly used in permutation based encodings.

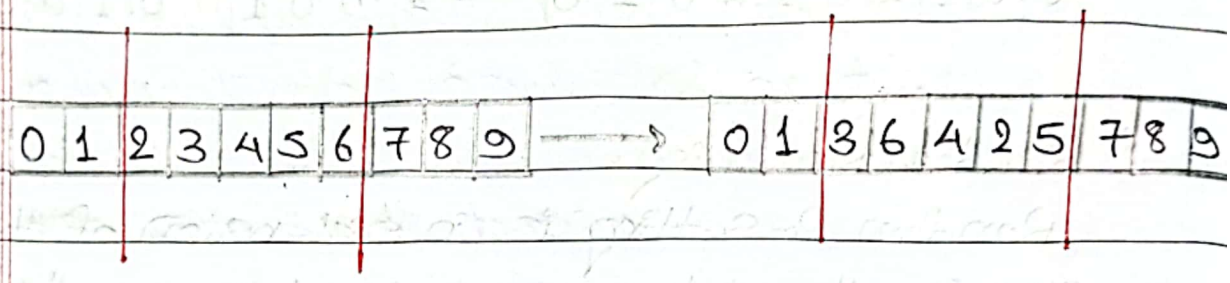


Date: _____

Page: _____

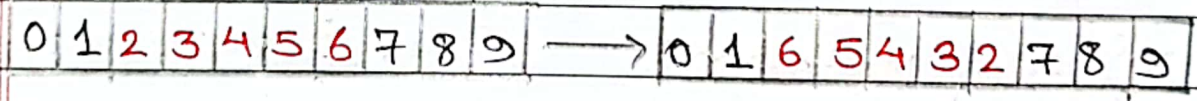
iv) Scramble Mutation:

Scramble mutation is popular with Permutation representations. In this, a subset of gene is chosen from the entire chromosome and their value are scrambled or shuffled randomly.



v) Inversion Mutation:

In inversion mutation, we select a subset of genes like in Scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset.



* Why Mutation:

→ Introduces Diversity in Solution Set,

→ Helps to stop Prematurse convergence.

→ Helps to move the solution from the domain of local maxima to Global maxima.

Date: .....

Page: .....

* Survivor Selection:

The Survivor selection Policy determines which individuals are to be kicked out and which are to be kept in the next generation. It is crucial in GA as it should ensure that the fitter individuals are not kicked out of the population, while at the same time diversity should be maintained in the population.

Some GAs employs Elitism. In simple terms, it means that the current fittest member of the population is always propagated to the next generation i.e. under any circumstances the fittest member of the current population can be replaced.

The easiest approach for survivor selection is to kick out random members out of the population, but such an approach frequently has convergence issues.

Some of the widely used strategies of survivor selection are as follow:

i) Age-Based Selection:

In Age-Based Selection, we don't have any idea of a fitness. It is based on the Premise that each individuals is allowed in the population for a finite generation where it is allowed to reproduce, after that, it is kicked out of the population no matter how good its fitness is.

Date: _____

Page: _____

Example:

For instance, in the following example, the age is the number of generation for which the individual has been in the population. The oldest members of the population i.e. P4 & P7 are kicked out of the population and the ages of the rest of the individuals are incremented by +1.

Age	Fitness value	chromosome
6	1	P_1
8	5	P_2
5	9	P_3
10	8	P_4
2	7	P_5
3	4	P_6
9	2	P_7
2	2	P_8
5	1	P_9
4	0	P_{10}

Existing Population

+

Fitness value	chromosome
8	C_1
9	C_2

OFFsprings

Date: .....

Page: .....

	Age	Fitness value	chromosome.
	7	1	P_1
	9	5	P_2
	6	9	P_3
*	0	8	C_1
	3	7	P_5
	4	4	P_6
*	0	9	C_2
	3	2	P_8
	6	1	P_9
	5	0	P_{10}

New population.

ii) Fitness Based Selection:

In this fitness based selection, the children or the offsprings tend to replace the least fit individuals in the population. The selection of the least fit individuals may be done using a variation of any of the selection policies described before - Tournament selection, fitness proportionate selection, etc.

* For example:

In the following figure, the children replaces the least fit individuals P_1 & P_{10} of the population. It is also to be noted that since

Date: .....

Page: .....

P1 & P9 have same fitness value, the decision to remove which individuals from the population is arbitrary.

Fitness value	chromosome
1	P_1
5	P_2
9	P_3
8	P_4
7	P_5
4	P_6
2	P_7
2	P_8
1	P_9
0	P_{10}

Existing population

Fitness value.

8	C_1
9	C_2

offsprings

*	Fitness value	chromosome
*	8	C_1
	5	P_2
	9	P_3
	8	P_4
	7	P_5
	4	P_6
	2	P_7
	2	P_8
	1	P_9
*	9	C_2

New population.

Date: .....

Page: .....

* Termination Condition:

The termination condition of a Genetic Algorithm is important in determining when a GA run will end. It has been observed that initially, the GA progresses very fast with better solutions coming in every few iterations, but this tends to slow down in the later stages where the improvements are very small. We usually want a termination condition such that our solution is close to the optimal at the end of the run.

Usually, we keep one of the following termination conditions:

i) When there has been no improvement in the population for X^{th} iterations.

ii) When we reach an absolute number of generations.

iii) When the objective function value has reached a certain pre-defined value.

For example, in a genetic algorithm we keep a counter which keeps track of the generation for which there has been no improvement in the population. Initially, we set this counter to zero. Each time when we don't generate off-springs which are better than the individuals in the populations, we increment the counter.

Date: _____

Page: _____

However, if the fitness of any of the off-springs is better, then we reset the counter to zero. and the algorithm terminates when the counter reaches a predetermined value.

Like other parameters of a GA, the termination condition is also highly problem specific and the GA designer should try out various options to see what suits his particular problem the be

Date: .....

Page: .....

* Mathematical Foundation of Genetic algorithm & building block hypothesis:

We will discuss about Schema, NFL theorem with Building block hypothesis

* Schema Theorem:

→ The Schema is a "similarity template".

→ Formally, it is a string over the alphabet = {0, 1, *} where, * is don't care & can take any values.

For eg:-

*10*1 could mean i.e. possible $2^2 = 4$ chromosome

① 01001

② 01011

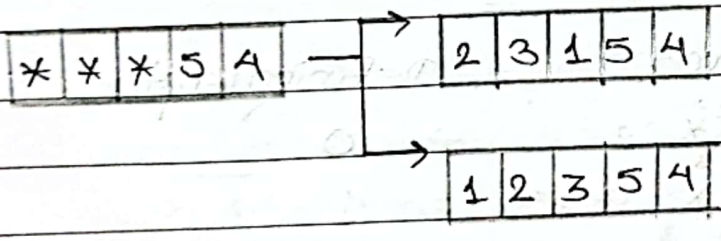
③ 11001

④ 11011

→ The schema is represented by H

H = *11*0*

→ In case of Permutation i.e,



i.e, 3 * 2 * 1 = 6 possibilities of chromosome

→ It should represent the solution which belongs to the solution set.

Date:

Page:

* Order of Schema:

→ Order of Schema is the number of specified fixed position in a ~~gene~~ : Schemata.

→ It is denoted by o(H).

i.e.

Schema	order
* * *	0
1 0 1	3
* 1 1	2
1 * *	1

* Defining length:

→ Defining length is the distance between the firs and last specific string position.

→ It can also defined as the distance between the two furthest fixed symbols in the genes or Schema.

Schema	Defining length
$^{0}*\ ^{1}*\ ^{2}*\ ^{3}*$ 0	0
$^{0}*\ ^{1}1\ ^{2}1\ ^{3}*$ (2-1)=1	1
$^{0}1\ ^{1}*\ ^{2}0\ ^{3}*$ (2-0)=2	2
$^{0}1\ ^{1}1\ ^{2}1\ ^{3}1$ (3-0)=3	3

→ It is denoted by δ(H).

Date:

Page:

* How many Schema are possible:

no. of Schema = $(n(A)+1)^L - 1$

where,

L = Length

A = {0, 1}, set of value of gene

for binary encoding = $3^L - 1$.

* Building Block Hypothesis:

→ Building Block are short defining length & low order schemata with the fitness value above the average fitness has high chances of surviving in the offspring.

→ The building block hypothesis say that such schemas serves as foundation for i) GA Secession &
ii) Adaptation in GA.

* Let see an example:

Schema 1 = 1 x x x 0 x 1 = H1 $\quad O(H_1) = 3,\ \delta(H_1) = 6$

Schema 2 = x x 10 x x x = H2 $\quad O(H_2) = 2,\ \delta(H_2) = 1$

① From H1 (1 x x x 0 x 1)

P1 = 1 00 1 | 0 11 $\quad$ C1 = 1 00 1 00 1 ? H1 survive

P2 = 1 1 01 | 0 0 1 $\quad$ C2 = 1 1 01 0 11

Date: .....

Page: .....

② From H_1 (1***0*1) generate P_1

From H_2 (**10***) generate P_2

i.e,

$P_1 = 1001|011$ $\quad C_1 = 1001100$ } only H_2 survives

$P_2 = 0110|100$ $\quad C_2 = 0110011$ }

(**)

V.U.Q

* Mathematical building block of Schemata & mathematical Process of Probability of Survival of Schemata.

= Effect of reproduction on Schemata:

$m(H,t)$ → no. of schemata H in population A(t), are m in generation t.

Also, $m(H,t+1)$ → what would be the no. of Schemata H in next generation in population A(t)

i.e, $m(H,t+1) = m(H,t) * \frac{f(H)}{\bar{f}}$

where,

$f(H)$ = Fitness of particular Schemata

$\bar{f} = \frac{\Sigma f}{n}$ (average fitness).

Date:

Page:

$$m(H, t+1) = m(H, t) \times \frac{f(H)}{\frac{\Sigma f}{n}}$$

$$= m(H, t) \times n \times \frac{f(H)}{\Sigma f}$$

now, probability of destroying of that schemata in crossover is

$$P_c = \frac{\delta(H)}{L-1}$$

i.e. long defining length has high probability of being destroyed.

so, Probability of survival of that schemata is.

$$1 - P_c \frac{\delta(H)}{L-1}$$

next, Probability of destroying of that schemata in mutation is.

$$P = P_m . O(H)$$

$p_m \rightarrow$ Probability of mutation.

So,

Probability of surviving schemata on mutation.

$$= 1 - P_m \cdot O(H)$$

Date:

Page:

nce,

overall survival of schemata over crossover & mutation is

$$m(H, t+1) \geq m(H,t) * n * \frac{f(H)}{\sum f_j} \left[1 - P_c \frac{\delta(H)}{L-1} - p_m o(H) \right]$$

* Schema Processing at work : An Example by Hand Revisited.

* problem : Find the optimal (best) value of x where, $f(x) = x^2$ & $x \in [0, 31]$

① Population size = 4

② Encoding = Binary encoding (use 5 bits to represent chromo since highest number is 31 in populatio

③ Fitness $(x) = f(x) = x^2$

= Initially, we take: 13 → 01101
24 → 11000
8 → 01000
19 → 10011

Date: string1=169 / 293

Page:

String Processing

String No.	x-value	Initial population (encoding)	Fitness $f(x)=x^2$	probability $\frac{f(x)}{\Sigma f}$	Expected Count $\frac{f_i}{\bar{f}}$
1	13	01101	169	14%	0.58
2	24	11000	576	49%	1.96
3	8	01000	64	6%	0.22
4	19	10011	361	31%	1.23
Sum			1170	100%	4
Average			293	25%	1
Max			576	49%	1.97

Actual Count		(based on experiment) Crossover	New population	Value	$f(x)=x^2$
1	2	4 (position of crossover with a string)	01100	12	144
2	1,3 (Use round off on Expected Count)	4 (1,2)	11001	25	625
0		2 (2,4)	11011	27	729
1	4 (The order is supposed to be result of experiment of parent selection)	2	10000	16	256
Sum	4				1754
Average	1				439
Max	2				729

Schema Processing

#Before Reproduction:

Schemata	(From initial population) string representation	Schema Average Fitness $f(H)$
H1: 1****	2,4	(576+361)/2 = 469
H2: *10**	2,3	(576+64)/2 = 320
H3: 1***0	2	576

Date:

Page:

*After Reproduction:

Expected Count	Actual Count	String representation (New population)
(1.96+1.23) = 3.20	3	2,3,4
(1.96+0.22) = 2.18	2	2,3
1.96	2	4

* Actual Count:

i) can be done by Round-off operation of Expected Co

ii) or using Formula:

i.e.,

$$m(H, t+1) = m(H, t) \cdot \frac{f(H)}{\bar{f}}$$

So,

① For, ~~shi~~ Schema 1 (H_1) = 1 * * * *

$O(H_1) = 1$ & $\delta(H_1) = 0$, String representation = 1

$m(H, t) = 2$

So,

$$m(H, t+1) = \frac{m(H, t) \cdot f(H)}{\frac{\Sigma f}{n}} = m(H, t) \times n \times \frac{f(H)}{\Sigma f}$$

$$m(H, t) \cdot \frac{f(H)}{\bar{f}} = 2 * \frac{469}{293} = 3.20$$

② H_2 = * 1 0 * *

$O(H_2) = 2$, $\delta(H_2) = 1$, String representation = 2,3

$m(H, t) = 2$

Date:

Page:

So

$$m(H, t+1) = m(H, t) \cdot \frac{f(H)}{F}$$

$$= 2 * \frac{320}{293}$$

$$= 2.18$$

$H_3 = 1 * * * 0$

$o(H_3) = 2$, $s(H_3) = 4$ & String representation = 2

$m(H, t) = 1$

So,

$$m(H, t+1) = m(H, t) \cdot \frac{f(H)}{F}$$

$$= 1 * \frac{576}{293}$$

$$= 1.96$$

Date:

Page:

* Imp questions?

i) How many schemata exists in a binary string of length 10:

:= Sol

Length = 10

We have two ways to find the no. of schemata is:

i) For binary string:

$3^L - 1$

$= 3^{10} - 1$

$= 59048$ schemata

ii) For any string:

$= (n(A)+1)^L - 1$

$= (2+1)^{10} - 1$ ($\because n(A) = \{0,1\}$ for binary)

$= 3^{10} - 1$

$= 59048$ schemata.

ii) How many schemata exists in a binary string of order 3 (i.e 3 position in chromosome are fixed) & length 10?

= Sol.

Length = 10

order = 3

We use combination which is:

$nC_r = {}^{10}C_3$

$= \frac{10!}{(10-3)!3!}$

Date: .....

Page: .....

$= \frac{.8 \times 5 \times 10}{3 \times 2}$

= 120 Schemata.

ii) How many Schemata exists with population size of 50 & length of schemata is 10? (MIN & MAX)

= Sol

Length = 10

Population size = 50

So,

The MINIMUM number of schemata is

= 10 (as there are 10 possible ways where any one position among 10th possible places can be fixed).

chromosome 1	1	*	*	*	*	*	*	*	*	*
chromosome 2	*	1	*	*	*	*	*	*	*	*
chromosome 3	*	*	1	*	*	*	*	*	*	*
chromosome 4	*	*	*	1	*	*	*	*	*	*
chromosome 5	*	*	*	*	1	*	*	*	*	*
chromosome 6	*	*	*	*	*	1	*	*	*	*
chromosome 7	*	*	*	*	*	*	1	*	*	*
chromosome 8	*	*	*	*	*	*	*	1	*	*
chromosome 9	*	*	*	*	*	*	*	*	1	*
chromosome 10	*	*	*	*	*	*	*	*	*	1

Date:

Page:

The Maximum possibilities is:

$= 2^{10}$ (Incase of binary Encoding)

So,

$= n^{10}$ (Incase of n-values)