UNIT - IV

4 Evolutionary Computing

4.1 Basic Evolutionary Processes

Q.1 What is general agreement that Darwinian evolutionary systems embody ?

Ans. : General agreement that Darwinian evolutionary systems embody :

1. One or more populations of individuals competing for limited resources,
2. The notion of dynamically changing populations due to the birth and death of individuals,
3. A concept of fitness which reflects the ability of an individual to survive and reproduce,
4. A concept of variational inheritance : offspring closely resemble their parents, but are not identical.

Q.2 How to represent the individuals (organisms) that make up an evolving population ?

Ans. : • A fairly general technique is to describe an individual as a fixed length vector of L features that are chosen presumably because of their (potential) relevance to estimating an individual's fitness.

• For example, individuals might be characterized by :

< hair color, eye color, skin color, height, weight >

• We could loosely think of this vector as specifying the genetic makeup of an individual, i.e., its genotype specified as a chromosome with five genes whose values result in an individual with a particular set of traits.

Q.3 What predictions might one make regarding the behavior of EV on this landscape ?

Ans. : Here are some possibilities :

1. EV will converge to a homogeneous fixed point as before. However, there are now four peaks of attraction. Which one it converges near will depend on the randomly generated initial conditions.
2. EV will converge to a stable fixed point in which the population is made up of sub-populations (species), each clustered around one of the peaks.
3. EV will not converge as before, but will oscillate indefinitely among the peaks.
4. The symmetry of the fitness landscape induces equal but opposing pressures to increment and decrement both feature values. This results in a dynamic equilibrium in which the average fitness of the population changes very little from its initial value

4.2 Evolutionary Systems as Problem Solvers

Q.4 What computation (if any) is EV performing ?

Ans. : • From an engineering perspective, systems are designed with goals in mind, functions to perform, and objectives to be met.

- Computer scientists design and implement algorithms for sorting, searching, optimizing, and so on.
- What is clear, however, is that even a system as simple as EV appears to have considerable potential for use as the basis for designing interesting new algorithms that can search complex spaces, solve hard optimization problems, and are capable of adapting to changing environments

4.3 Canonical Evolutionary Algorithm

Q.5 How to transform incremental EV into a batch system ?

Ans. : • It is easy to transform incremental EV into a batch system by explicitly maintaining two populations, one for m parents and one for n offspring.

• The two populations interact as follows :

For I = 1 to n Do :

Randomly select an individual from the parent population to produce a child as before, and add the child to the offspring population.

For I = 1 to n Do :

Force offspring I to compete for space in the parent population as before.

Q.6 What is evolutionary programming ?

Ans. : • Evolutionary programming is one of the four major evolutionary algorithm paradigms. It is similar to genetic programming, but the structure of the program to be optimized is fixed, while its numerical parameters are allowed to evolve.

• For EP, like Gas, there is an underlying assumption that a FITNESS landscape can be characterized in terms of variables, and that there is an optimum solution in terms of those variables.

The basic EP method involves 3 steps :

1. Choose an initial POPULATION of trial solutions at random. The number of solutions in a population is highly relevant to the speed of OPTIMIZATION, but no definite answers are available as to how many solutions are appropriate (other than >1) and how many solutions are just wasteful.
2. Each solution is replicated into a new POPULATION. Each of these OFFSPRING solutions are mutated according to a distribution of MUTATION types, ranging from minor to extreme with a continuum of mutation types between. The severity of MUTATION is judged on the basis of the functional change imposed on the PARENTS.
3. Each OFFSPRING solution is assessed by computing it's FITNESS. Typically, a stochastic tournament is held to

determine N solutions to be retained for the POPULATION of solutions, although this is occasionally performed deterministically. There is no requirement that the POPULATION SIZE be held constant, however, nor that only a single OFFSPRING be generated from each PARENT.

Q.7 What is difference between EVOLUTIONARY PROGRAMMING (EP) and EVOLUTION STRATEGY (ES) ?

Ans. : The main differences between EP and ES are as follows :

1. **Selection** : EP typically uses STOCHASTIC SELECTION via a tournament. Each trial SOLUTION in the POPULATION faces competition against a preselected number of opponents and receives a "win" if it is at least as good as its opponent in each encounter. SELECTION then eliminates those SOLUTIONS with the least wins. In contrast, ES typically uses deterministic SELECTION in which the worst SOLUTIONS are purged from the POPULATION based directly on their function evaluation.
2. **Recombination** : EP is an abstraction of EVOLUTION at the level of reproductive POPULATIONs and thus no RECOMBINATION mechanisms are typically used because RECOMBINATION does not occur between SPECIES. In contrast, ES is an abstraction of EVOLUTION at the level of INDIVIDUAL behavior. When self-adaptive information is incorporated this is purely GENETIC information and thus some forms of RECOMBINATION are reasonable and many forms of RECOMBINATION have been implemented within ES. Again, the effectiveness of such operators depends on the problem at hand.

Q.8 How EVOLUTIONARY PROGRAMMING differs from genetic algorithm ?

Ans. : There are two important ways in which EP differs from GAs.

- First, there is no constraint on the representation. The typical GA approach involves encoding the problem solutions as a string of representative tokens, the GENOME. In EP, the representation follows from the problem. A neural network can be represented

in the same manner as it is implemented, for example, because the MUTATION operation does not demand a linear encoding.

- Second, the MUTATION operation simply changes aspects of the solution according to a statistical distribution which weights minor variations in the behavior of the OFFSPRING as highly probable and substantial variations as increasingly unlikely. Further, the severity of MUTATIONS is often reduced as the global optimum is approached. There is a certain tautology here : if the global optimum is not already known, how can the spread of the mutation operation be damped as the solutions approach it ? Several techniques have been proposed and implemented which address this difficulty, the most widely studied being the "Meta-Evolutionary" technique in which the variance of the mutation distribution is subject to mutation by a fixed variance mutation operator and evolves along with the solution.

END...

UNIT - V

5 Genetic Algorithm

5.1 Basic Concepts and Working Principle

Q.1 What is Genetic algorithm ? List components of GA. Explain Psuedocode of Genetic algorithm.

Ans. : • As early as 1962, John Holland's work on adaptive systems laid the foundation for later developments. By the 1975, the publication of the book *Adaptation in Natural and Artificial Systems*, by Holland and his students and colleagues.

- Early to mid-1980s, genetic algorithms were being applied to a broad range of subjects. In 1992 John Koza has used genetic algorithm to evolve programs to perform certain tasks. He called his method genetic programming.
- **Genetic Algorithm (GA)** is a search technique used in computing to find true or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics.
- Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection and recombination.
- GA tries to perform an intelligent search for a solution from a nearly infinite number of possible solutions. It works with coding variables. Genetic algorithm and their variants are sometimes referred to as methods of population based optimization.
- Genetic algorithms are also categorized as global search heuristics. Actually they are a particular class of evolutionary

algorithms which uses techniques inspired by evolutionary biology such as inheritance, mutation, selection and crossover.

- A typical genetic algorithm requires two things to be defined :
  1. Genetic representation of the solution domain.
  2. Fitness functions to evaluate the solution domain.

Components of Genetic Algorithm

- Genetic algorithm consists of encoding schemes, fitness evaluations, parent selection, crossover operators, and mutation operators.
- After an initial population is randomly generated, the algorithm evolves the through three operators :
  1. Selection which equates to survival of the fittest;
  2. Crossover which represents mating between individuals;
  3. Mutation which introduces random modifications.
- The Genetic algorithm starts off with an initial population P(0) and it is followed by selection, crossover and mutation processes. These processes are shown in Fig. Q.1.1.

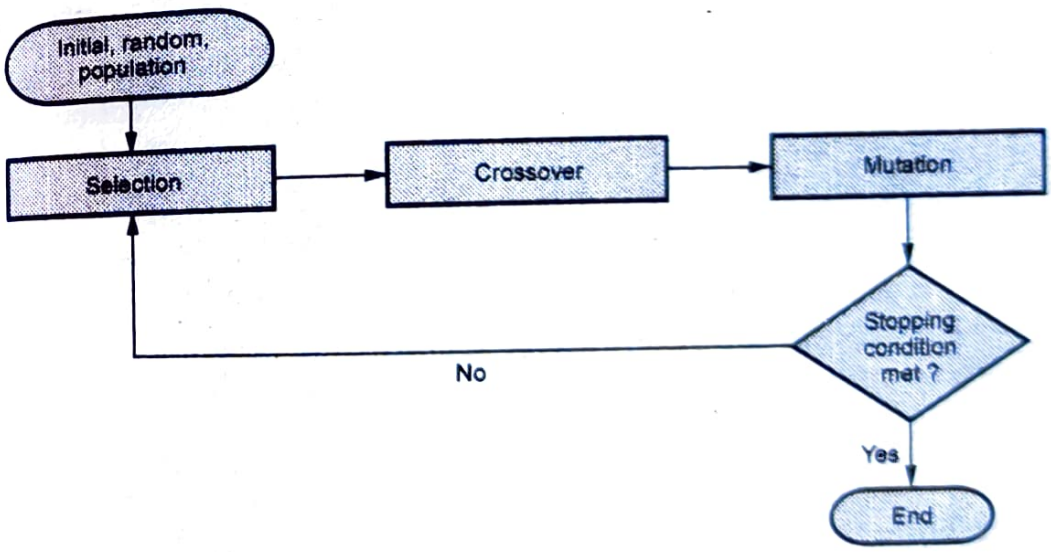


Fig. Q.1.1 One generation of GA process

Psuedocode of Genetics Algorithm

1. Choose the initial population of individuals
2. Evaluate the fitness of each individual in population
3. Repeat until termination condition satisfied :
   3a. Selection : Select the individuals with greater fitness for reproduction
   3b. Crossover : Breed new individuals through crossover
   3c. Mutation : Apply probabilistic mutation on new individuals
   3d. Form a new population with these offsprings.
4. Terminate

- Genetic algorithm starts with the current population. Selection is applied to the current population to create an intermediate population. Then recombination and mutation are applied to the intermediate population to create the next population. The process of going from the current population to the next population constitutes one generation in the execution of a genetic algorithm.

Q.2 Why use Genetic algorithm ?

Ans. : • Genetic algorithms evaluate the target function to be optimized at some randomly selected points of the definition domain. Genetic algorithms can be used when no information is available about the gradient of the function at the evaluated points. The function itself does not need to be continuous or differentiable.

- Genetic algorithms can still achieve good results even in cases in which the function has several local minima or maxima.
- Genetic algorithms are stochastic search algorithms which act on a population of possible solutions. They are loosely based on the mechanics of population genetics and selection.
- Genetic Algorithms allow you to explore a space of parameters to find solutions that score well according to a "fitness function".
- Genetic Programming takes genetic algorithms a step further, and treats programs as the parameters. For example, you would

breeding path finding algorithms instead of paths, and your fitness function would rate each algorithm based on how well it does.

Q.3 Define : a. Cell b. Genotype c. Phenotype

Ans. : a. Cell : Animal or human cell is a complex of many small factories that work together. The center of all this is the cell nucleus. The genetic information is contained in the cell nucleus.

b. Genotype for a particular individual, the entire combination of genes is called genotype.

c. Phenotype the phenotype describes the physical aspect of decoding a genotype to produce the phenotype.

Q.4 With the neat flowchart explain operation of simple genetic algorithms.

Ans. : 1. **[Start]** Generate random population of n chromosomes (suitable solutions for the problem).

2. **[Fitness]** Evaluate the fitness **f(x)** of each chromosome **x** in the population.

3. **[New population]** Create a new population by repeating following steps until the new population is complete.

   a. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).

   b. **[Crossover]** with a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.

   c. **[Mutation]** with a mutation probability mutate new offspring at each locus (position in chromosome).

   d. **[Accepting]** Place new offspring in a new population.

4. **[Replace]** Use new generated population for a further run of algorithm.

5. **[Test]** If the end condition is satisfied, stop and return the best solution in current population.

6. **[Loop]** Go to step 2.

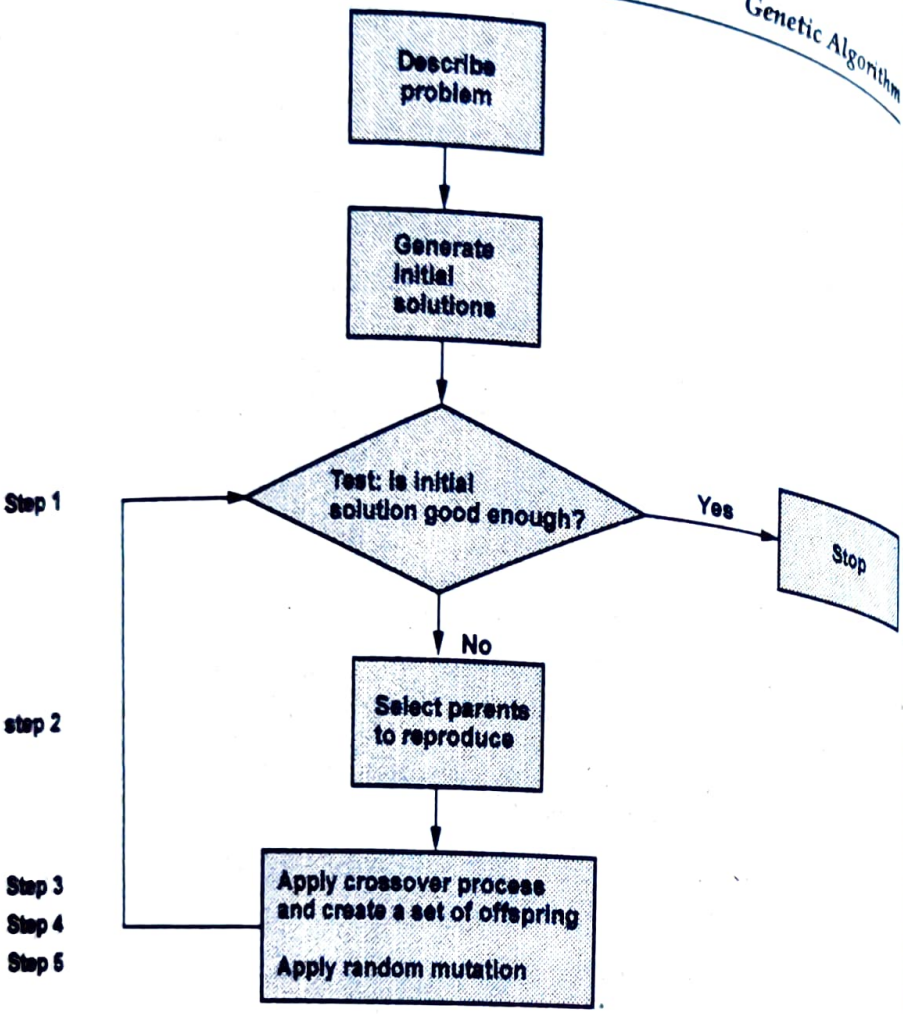


Fig. Q.4.1 Flow diagram of the genetic algorithm process

GA Steps

Step 1 : Set population size and probability

Step 2 : Define fitness function

Step 3 : Generate initial population

Step 4 : Calculate fitness for each chromosome

Step 5 : Mating of chromosomes

Step 6 : Create offspring-crossover and mutation

Step 7 : Offspring in new population

Step 8 : Repeat Step 5 until new = initial population

Step 9 : Replace initial population with new

Step 10 : To Step 4 and repeat until criteria achieved

Q.5 What are limitations on genetic algorithms ?

Ans. : • GAs are not guaranteed to find the global optimum solution to a problem.

• GAs are an extremely general tool and they have no specific way for solving particular problems.

• GAs are usually used when everything else is failed or when we don't have enough knowledge of the search space.

• Even when such specialized techniques exist, it is often interesting to hybridise them with a GA in order to possibly gain some improvements.

Q.6 Explain advantages of Genetic algorithm.

Ans. : • Genetic Algorithm is a stochastic algorithm.

• Randomness as an essential role in both selection and reproduction phases.

• Genetic algorithms always consider a population of solutions. A population base algorithm is also very amenable for parallelization.

• There is no particular requirement on the problem before using genetic algorithms, so it can be applied to resolve any problem (optimization).

• GAs are a new field and parts of the theory have still to be properly established. We can find almost as many opinions on GAs as there are researchers in this field.

Q.7 Explain basic operator used in Genetic algorithm.

Ans. : • A genetic algorithm maintains a population of candidate solutions for the problem at hand and makes it evolve by iteratively applying a set of stochastic operators. The simplest form of genetic algorithm involves three types of operators : selection, crossover and mutation.

1. **Selection :** Selection replicates the most successful solutions found in a population at a rate proportional to their relative

quality. This operator selects chromosomes in the population for reproduction. The fitter the chromosome, the more times it is likely to be selected to reproduce.

2. **Crossover** : This operator randomly chooses a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring. For example, the strings 10000100 and 11111111 could be crossed over after the third locus in each to produce the two offspring 10011111 and 11100100. The crossover operator roughly mimics biological recombination between two single chromosome (haploid) organisms.

3. **Mutation** : Mutation randomly perturbs a candidate solution. This operator randomly flips some of the bits in a chromosome. For example, the string 00000100 might be mutated in its second position to yield 01000100. Mutation can occur at each bit position in a string with some probability, usually very small (e.g., 0.001).

5.2 Genetic Operators

Q.8 What are operators in genetic algorithms ? List and explain GA operators in brief.

Ans. : • Generation of successors is determined by a set of operators that recombine and mutate selected members of the current population. Operators correspond to idealized versions of the genetic operations found in biological evolution.
(Refer Fig. Q.8.1)

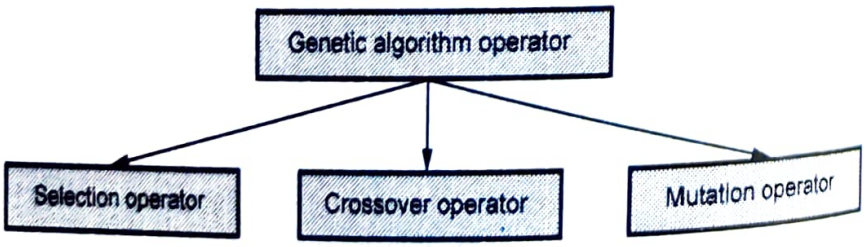


Fig. Q.8.1

- Genetic operator is the operator for generating new gene based on fitness of each individual.

Selection Operator

- Key idea : Give preference to better individuals, allowing them to pass on their genes to the next generation.
- It determines which parents participate in producing offspring for the next generation. The goodness of each individual depends on its fitness.
- Fitness may be determined by an objective function or by a subjective judgement.
- Selection replicates the most successful solutions found in a population at a rate proportional to their relative quality.
- A common selection method in genetic algorithm is fitness proportionate selection, in which the number of times an individual is expected to reproduce is equal to its fitness divided by the average of fitness in the population.
- Common selection methods used in GAs are,

a. Fitness proportionate selection

b. Rank selection

c. Tournament selection

Q.9 What are types of crossover and mutation techniques ?

Ans. : • A binary variation operator is called recombination or crossover. A method of mixing good solutions to produce better ones is called crossover.

- Crossover means choosing a random position in the string (say, after 2 digits) and exchanging the segments either to the right or to the left of this point with another string partitioned similarly to produce two new offspring.

- Crossover produces two new offspring from two parent st by copying selected bits from each parent. Bit at position "i each offspring is copied from the bit at position "I" in one o two parents. The choice which parent contributes bit "I determined by an additional string, called **cross-over mask**.
- In the crossover operator, new strings are created by exchang information among strings of the mating pool.
- The primary objective of the recombination operator is emphasize the good solutions and eliminate the bad solutions a population, while keeping the population size constant.
- "Selects The Best, Discards The Rest". "Recombination" is differe from "Reproduction".
- Cross over can be rated complicated and very depends encoding of the chromosome. Specific crossover made for specific problem can improve performance of the genet algorithm.

Steps of crossover :

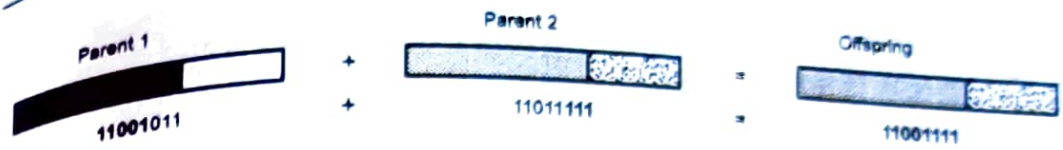
1. The reproduction operator selects at random a pair of tw individual strings for the mating.
2. A cross site is selected at random along the string length.
3. Finally, the position values are swapped between the two strings following the cross site.

Crossover Types

1. Single-point crossover,
2. Two-point crossover,
3. Uniform crossover

One point crossover

- One point crossover is the most basic crossover operator. Crossover point on the genetic code is selected at random and two parent chromosomes are interchanged at this point. Binary string from beginning of chromosome to the crossover point is copied from one parent, the rest is copied from the second parent.



- Example :

Parent 1 : X X | X X X X X

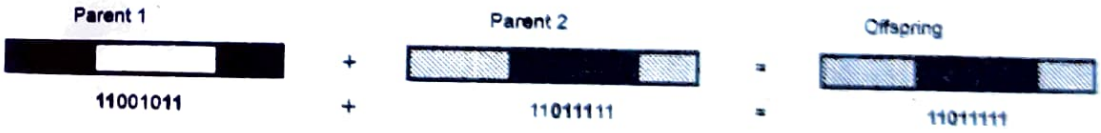
Parent 2 : Y Y | Y Y Y Y Y

Offspring 1 : X X Y Y Y Y Y

Offspring 2 : Y Y X X X X X

Two Point Crossover

- Two-point crossover is very similar to single-point crossover except that two cut-points are randomly generated instead of one.
- Two point crossover : Two crossover points are selected and the part of the chromosome string between these two points is then swapped to generate two children. Binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent.



- Example :

Parent 1 : X X | X X X | X X

Parent 2 : Y Y | Y Y Y | Y Y

Offspring 1 : X X Y Y Y X X

Offspring 2 : Y Y X X X Y Y

Uniform Crossover

- In uniform crossover, a value of the first parent's gene is assigned to the first offspring and the value of the second parent's gene is to the second offspring with a probability value p_c.
- With probability p_c the value of the first parent's gene is assigned to the second offspring and the value of the second parent's gene is assigned to the first offspring.
- Example :

  Parent 1 : X X X X X X X

  Parent 2 : Y Y Y Y Y Y Y

  Offspring 1 : X Y X Y Y X Y

  Offspring 2 : Y X Y X X Y X

- Crossover between 2 good solutions MAY NOT ALWAYS yield a better or as good a solution. Since parents are good, probability of the child being good is high. If offspring is not good (poor solution), it will be removed in the next iteration during "Selection".
- One site crossover is more suitable when string length is small while two site crossover is suitable for large strings.
- **Crossover example**

  Parent A 011011

  Parent B 101100

- Mate the parents by splitting each number as shown between the second and third digits (position is randomly selected).

  01*1011 10*1100

- Now combine the first digits of A with the last digits of B and the first digits of B with the last digits of A. This gives you two new offspring.

  011100

  101011

- If these new solutions or offspring, are better solutions than the parent solutions, the system will keep these as more optimal solutions and they will become parents. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

Matrix Crossover

- Some real problems are naturally suitable for two-dimensional representation. If this kind of problems is to be solved by genetic algorithms, then each possible solution can very conveniently and naturally be conceptually represented as a two-dimensional table.
- Two-dimensional substring crossover generates two offspring chromosomes by choosing only one of the two crossover strategies (horizontal or vertical).
- Alternatively, the two-dimensional crossover operator can be easily modified to generate four offspring chromosomes from a pair of parents by executing the horizontal and the vertical crossovers at the same time.
- The new offspring chromosomes that result from executing the crossover operation may become infeasible for some application problems. Refer Fig. Q.9.1.

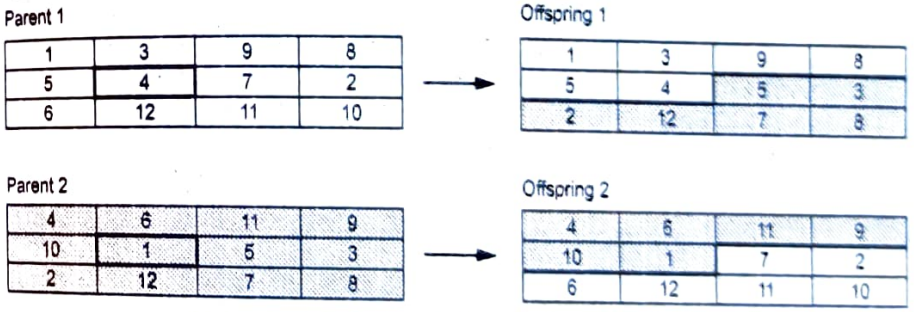


Fig. Q.9.1 Horizontal substring crossover

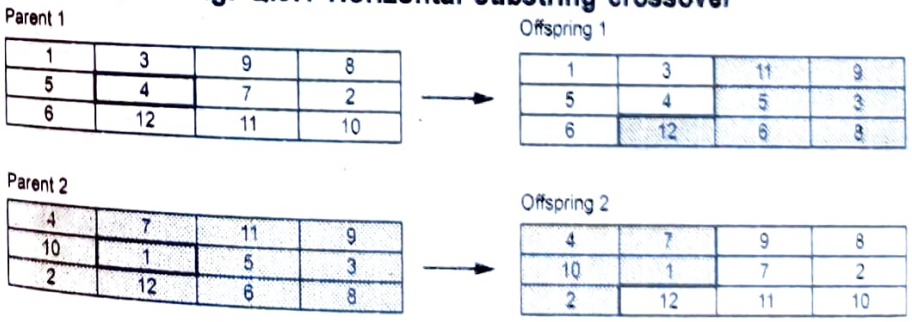


Fig. Q.9.2 Vertical substring crossover

Mutation Operator

- It helps to increase searching power. In the mutation process, each element in an individual is changed with a small probability of p_m. With some low probability, a portion of the new individuals will have some of their bits flipped.
- Main purpose is to maintain diversity within the population and inhibit premature convergence. Mutation alone induces a random walk through the search space.
- The mutation rate is usually kept low so good chromosomes obtained from crossover are not lost. Mutation increases diversity in the population and thus allows the algorithm to search more distinct points.
- The mutation depends on the encoding as well as the crossover. For example when we are encoding permutations, mutation could be exchanging two genes.
- It is the process by which a string is deliberately changed so as to maintain diversity in the population set. **Mutation probability** determines how often the parts of a chromosome will be mutated.
- Mutation adds new information in a random way to the genetic search process and ultimately helps to avoid getting trapped at local optima. It is an operator that introduces diversity in the population whenever the population tends to become homogeneous due to repeated use of reproduction and crossover operators.
- Mutation may cause the chromosomes of individuals to be different from those of their parent individuals.
- If crossover is supposed to exploit the current solution to find better ones, mutation is supposed to help for exploration of the whole search space. Mutation in a way is the process of randomly disturbing genetic information. They operate at the bit level; when the bits are being copied from the current string to the new string, there is probability that each bit may become mutated.

- The need for mutation is to create a point in the neighborhood of the current point, thereby achieving a local search around the current solution. The mutation is also used to maintain diversity in the population. For example, the following population having four eight bit strings may be considered :

  01101011

  00111101

  00010110

  01111100

- It can be noticed that all four strings have a 0 in the left most bit position. If the true optimum solution requires in that position, then neither reproduction nor crossover operator described above will be able to create 1 in that position.

Q.10 What is encoding ? When to select encoding method ? Explain type of binary encoding methods.

Ans. : • Encoding is the process of representing the solution in the form of a string that conveys the necessary information. It is a process of representing individual genes. The process can be performed using bits, numbers, trees, array or any other objects.

- Just as in a chromosome, each gene controls a particular characteristic of the individual; similarly, each bit in the string represents a characteristic of the solution.
- When choosing an encoding method rely on the following key ideas

1. Use a data structure as close as possible to the natural representation.
2. Write appropriate genetic operators as needed.
3. If possible, ensure that all genotypes correspond to feasible solutions.
4. If possible, ensure that genetic operators preserve feasibility

Binary Encoding

- Most common method of encoding. Chromosomes are strings of 1s and 0s and each position in the chromosome represents a particular characteristic of the problem. The length of the string is usually determined according to the desired solution accuracy.

  Chromosome A 110100011010

  Chromosome B 011111111100

- **Octal encoding** : This encoding uses string made up of octal numbers 0 to 7.

  Chromosome A 03467216

  Chromosome B 15723314

- **Hexadecimal encoding** : This encoding uses string made up of hexadecimal numbers 0 to 9 and A to F.

  Chromosome A 9CE7

  Chromosome B 3DBA

Permutation Encoding

- Useful in ordering problems such as the Traveling Salesman Problem (TSP). Example : In TSP, every chromosome is a string of numbers, each of which represents a city to be visited.
- Permutation encoding is used in scheduling and ordering problems. Every chromosome is a string of numbers, which represents numbers in a sequence. The kind of encoding usually needs to make some corrections after the crossover and mutation operating procedures because any transformation might create an illegal situation.

  Chromosome A 1 5 3 2 6 4 7 9 8

  Chromosome B 8 5 6 7 2 3 1 4 9

- **Example of problem** : Traveling Salesman Problem (TSP).
- **The problem** : There are cities and given distances between them. Traveling salesman has to visit all of them, but he does not to travel very much. Find a sequence of cities to minimize

traveled distance. **Encoding :** Chromosome says order of cities, in which salesman will visit them.

Value Encoding

- Used in problems where complicated values, such as real numbers, are used and where binary encoding would not suffice. It is good for some problems, but often necessary to develop some specific crossover and mutation techniques for these chromosomes.
- **Example of problem :** Finding weights for neural network.
- **The problem :** There is some neural network with given architecture. Find weights for inputs of neurons to train the network for wanted output.
- **Encoding :** Real values in chromosomes represent corresponding weights for inputs.

Chromosome A **1.2324 5.3243 0.4556 2.3293 2.4545**

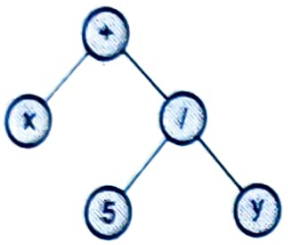
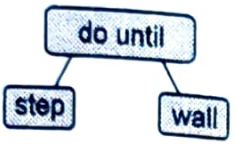
Chromosome B **ABDJEIFJDHDIERJFDLDFLFEGT**

Chromosome C **(back), (back), (right), (forward), (left)**

Tree Encoding

- In **tree encoding** every chromosome is a tree of some objects, such as functions or commands in programming language. It is used in genetic programming.
- **Example of problem :** Finding a function from given values.
- **The problem :** Some input and output values are given. Task is to find a function, which will give the best (closest to wanted) output to all inputs.

- **Encoding** : Chromosome is functions represented in a tree.

Chromosome A	Chromosome B
Chromosome A  (+ x (/ 5 y))	Chromosome B  (do_until step wall)

Q.11 What is Fitness function ?

Ans. : • Fitness is an important concept in genetic algorithms. The fitness of a chromosome determines how likely it is that it will reproduce. Fitness is usually measured in terms of how well the chromosome solves some goal problem. Fitness can also be subjective (aesthetic). E.g., if the genetic algorithm is to be used to sort numbers, then the fitness of a chromosome will be determined by how close to a correct sorting it produces.

- A fitness function quantifies the optimality of a solution (chromosome) so that particular solution may be ranked against all the other solutions. A fitness value is assigned to each solution depending on how close it actually is to solving the problem.
- Ideal fitness function correlates closely to goal plus quickly computable.
- Example : In TSP, f(x) is sum of distances between the cities in solution. The lesser the value, the fitter the solution is.
- The performance of the individual strings is measured by a fitness function. A fitness function is a problem specific user defined heuristic. After each iteration, the members are given a performance measure derived from the fitness function and the "fittest" members of the population will propagate the next iteration.

Fitness = F_i, Hit = λ_i, Survival = ϕ_i, Death= δ_i

$$F_i = 2\lambda_i - \delta_i + \sum \phi_i$$

- The fitness function is defined over the genetic representation and measures the *quality* of the represented solution. The fitness function is always problem dependent.
- For instance, in the knapsack problem we want to maximize the total value of objects that we can put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object and the value of the bit (0 or 1) represents whether or not the object is in the knapsack.
- Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The *fitness* of the solution is the sum of values of all objects in the knapsack if the representation is valid or 0 otherwise. In some problems, it is hard or even impossible to define the fitness expression; in these cases, interactive genetic algorithms are used.

5.3 Traditional Algorithm vs Genetic Algorithm

Q.12 Explain difference between genetic algorithm and Traditional algorithm.

Ans. : 1. Genetic algorithms a coded form of the function values i.e. parameter set, rather than with the actual values themselves. For example, if we want to find the minimum of the function $f(x) = x^3 + x^2 + 5$, the GA would not deal directly with x or y values, but with strings that encode these values. For this case, strings representing the binary x values should be used.

2. Genetic algorithms use a set or population of points to conduct a search, not just a single point on the problem space. This gives GAs the power to search noisy spaces littered with local optimum points. Instead of relying on a single point to search through the space, the GAs looks at many different areas of the problem space at once, and uses all of this information to guide it.

3. Genetic algorithms use only payoff information to guide themselves through the problem space. Many search techniques need a variety of information to guide themselves. Hill climbing methods require derivatives, for example. The only information a GA needs is some measure of fitness about a point in the space. Once the GA knows the current measure of "goodness" about a point, it can use this to continue searching for the optimum.
4. GAs are probabilistic in nature, not deterministic. This is a direct result of the randomization techniques used by GAs.
5. GA is inherently parallel. Here lies one of the most powerful features of genetic algorithms. GA's, by their nature, are very parallel, dealing with a large number of points simultaneously.

Parameters	Genetic Algorithms	Traditional Methods
Work with	Coding of parameter set	Parameters directly
Use information	Payoff i.e. objective function	Payoff plus derivatives etc.
Rules	Probabilistic	Fully deterministic
Search	A population of points	A population of points a single point

5.4 Simple GA and General Genetic Algorithm

Q.13 Write short note on general GA.

Ans. : • General genetic algorithm starts with a randomly chosen population which is made up of binary series. The criterion function helps us determine their quality and based on this information, the number of copies which each individual subject will have at the construction of the next generation is set.

- The acquired copies cross-over randomly between themselves and a random selection is made of the genetic string length which will cross-over or be switched.
- **Initialization** : Initially many individual solutions are randomly generated to form an initial population. Nature of problem decides population size. It covers entire search space. The

solutions may be seeded in areas where optimal solutions are likely to be found.

- **Selection :** During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a *fitness-based* process, where fitter solutions are typically more likely to be selected. Most functions are stochastic and designed so that a small proportion of less fit solutions are selected.

- **Reproduction :** For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each child and the process continues until a new population of solutions of appropriate size is generated.

- **Termination :** This generational process is repeated until a termination condition has been reached. Common terminating conditions are :

a. A solution is found that satisfies minimum criteria.

b. Fixed number of generations reached.

c. Allocated budget reached.

d. The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results.

e. Manual inspection.

5.5 Schema Theorem

Q.14 What is schema and schema theorem?

Ans. : • A schema is a similarity template describing a subset of strings with similarities at certain string positions. Other name for a schema is a building block of a chromosome.

- A schema is a set of binary strings that match the template for schema H.
- The notion of schemata and the Schema theorem provide the theoretical background why genetic algorithms work so well in practice.
- To describe building blocks of **binary** chromosomes, the following **three letter** alphabet is used : **0, 1,** and * called **a do not care symbol.**
- 1 or 0 at any position of a building block means that the chromosome must have the same bit at that position for it to contain the building block.
- The "do not care" - "*" symbol at any position in the building block means that the value of the chromosome bit at this position is irrelevant to determining whether or not the chromosome contains the building block.
- Schema Examples : The schema H = 10*1* represents the set of binary strings 10010, 10011, 10110, 10111.
- The string '10' of length l = 2 belongs to 2l = 22 different schemas **, *0, 1*, 10.
- **Schema theorem** : Let **s**(t) be the number of chromosomes matching a schema **s** in generation t, let $\phi(s, t)$ be the growth rate of schema in generation t due to selection and let $\varepsilon(s, t)$ be the decay rate of the schema in generation t due to disruption by crossover and mutation. Then the following holds for every schema s in the population :

$$s(t + 1) \geq s(t) \cdot \phi(s, t) \cdot \varepsilon(s, t)$$

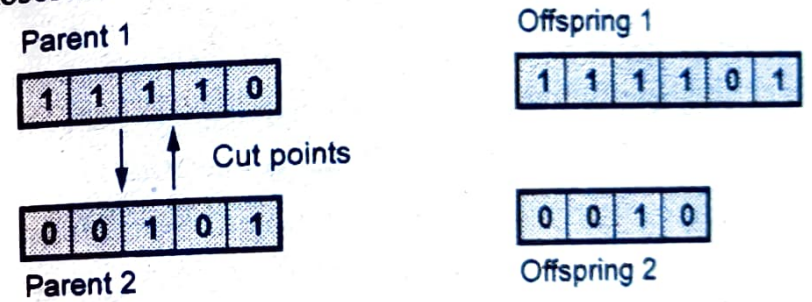
- The schema theorem states that the number of chromosomes matching a schema will increase due to the effect of selection if the schema has above-average fitness, but it will decrease as a result of the disruptive effects of crossover and mutation.

5.6 Classification of Genetic Algorithm

Q.15 Discuss Classification of genetic algorithm.

Ans. : 1. Messy Genetic Algorithms (mGA)

- It is developed as an alternative to standard genetic algorithms. Each bit is labeled with its position.
- A chromosome does not have to contain a value for each position and a given position in a chromosome can have more than one value.
- Each bit in a chromosome is represented by a pair of numbers : the first number represents the position within the chromosome and the second number is the bit value.
- It use the standard mutation operation. But, instead of crossover, they use splice and cut operations. Two chromosomes can be spliced together by simply joining one to the end of other. The cut operator splits one chromosome, into two smaller chromosomes.



- The process for running this genetic algorithm is as follows :
  1. Produce a random population of chromosomes. We will start with 100.
  2. Determine a score for each chromosome by playing its strategy against a number of opponents.
  3. Select chromosomes for the population to reproduce.
  4. Replace the previous generation with new population produced by reproduction.
  5. Return to step 2.

2. Adaptive Genetic Algorithm

- A matrix formulation for an adaptive genetic algorithm is developed using mutation matrix and crossover matrix.
- In the classic genetic algorithm, the involved genetic operators, such as crossover and mutation, work at an a priori, constant probability. Different crossover and mutation rates can, however, traverse different search directions in the state space, thus affecting the performance of the applied genetic algorithm.
- Crossover occurs only with some probability ρ_c (the crossover probability or crossover rate). When the solutions are not subjected to crossover, they remain unmodified. Notable crossover techniques include the single-point, the two-point, and the uniform types.
- Mutation involves the modification of the value of each 'gene' of a solution with some probability P_m (the mutation probability).

5.7 Holland Classifier Systems

Q.16 What is classifier system? Explain Holland classifier systems.

Ans. : • A Classifier System (CS) is a machine learning system that learns syntactically simple string rules, called classifiers.

- The learning classifier system, introduced by Holland and Reitman is a machine learning system which possesses the salient properties needed to learn in the shape optimization domain.
- A classifier system derives its name from its ability to learn to classify messages from the environment into general sets and is similar to a control system in many respects.
- Fig. Q.16.1 shows the interactions between the classifier system and the environment.

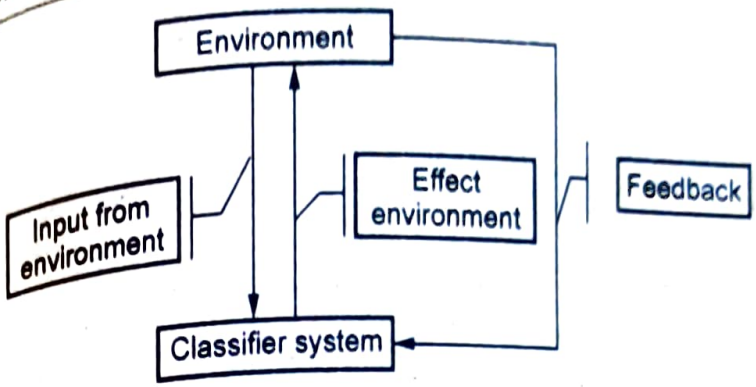


Fig. Q.16.1 Interactions between Classifier System and Environment

- In this computational model, a learning classifier has three major components : a performance system, an apportionment of credit procedure, and a rule discovery system.
- The classifier system receives information about the environment, performs internal processing and then effects the environment.
- It then uses feedback about the effect on the environment to learn from the experience. This arrangement has the classifier system in learning mode, because the classifier system is utilizing the feedback to learn from experience.
- Conversely, if no feedback is provided, the classifier system is in application mode. Application mode is utilized after sufficient learning is accomplished.
- The performance system consists of a message list and a collection of rules. The rules represent the knowledge base of the system.
- At any time in the processing, the collection of rules is the system's best approximation to the knowledge the system is processing. However, all communication within the system involves messages.
- Input from the environment is formatted by the sensors into messages and placed on the message list. The rules specify conditions for their execution and produce messages or perhaps output as effectors. The rules may be thought of as if-then statements.

- An input message is processed by matching the first portion of rules, including effector conditions. That is an effect or appears in the rule collection as other rules. The difference is that it produces output rather than another message.
- One or more rules may match the message. One is selected and if it is not an effector then it outputs a new message to the list. This may be repeated several times before the message placed on the list matches an effectors condition.

Q.17 Explain Bucket Brigade Algorithm.

Ans. : • It is developed by Holland for the apportionment of credits that relies on the model of a service economy, consisting of two main components : auction and a clearing house.

- The environment as well as the classifiers post messages.
- Each classifier maintains a bank account that measures its strength. Classifiers that match a posted string, make a bid proportial to their strength. Usually, the highest bidding classifier is selected to post its message (other, more parallel schemes are also used)
- The auction permits appropriate classifiers to post their messages. Once a classifier is selected for activation, it must clear its payments through a clearing house paying its bid to other classifiers or the environment for matching messages rendered.
- A matched and activated classifier sends its bid to those classifiers responsible for sending messages that matched the bidding classifiers conditions. The sent bid-money is distributed in some manner between those classifiers.
- Rules that cooperate with a classifier are rewarded by receiving the classifiers bid, the last classifier in a chain receives the environmental reward, all the other classifiers receive the reward from their predecessor.

- A classifier's strength might be subject to taxation. The idea that underlies taxation is to punish inactive classifiers : $T_i(t) := C_{tax} * S_i(t)$
- The strength of a classifier is updated using the following equation :

$$S_i\ (t+1) = S_i(t) - P_i(t) - T_i(t) + R_i(t)$$

- A classifier bids proportional to its strength : $B_i = C_{bid} * S_i$.
- Genetic algorithms are used to evolve classifiers. A classifiers strength defines its fitness, fitter classifiers reproduce with higher probability and binary string mutation and crossover operators are used to generate new classifiers.
- Newly generated classifiers replace weak, low strength classifier
- The following is the basic algorithm using the bucket brigade algorithm :

0. Set initial strength to all classifiers (rules). Clear list and append all input messages to list. Set all classifiers to not active.
1. Set to active all classifiers that match messages on the list to active and clear list.
2. For each classifier that is marked active calculate a bid quantity
3. Choose stochastically with a probability related to the bid quantity classifiers to add new messages to the list. (Each message is tagged with the classifier that added it to make possible 4.)
4. Each classifier that has successfully posted a new message pays the classifier that posted the message that caused it to be active its payment quantity, which is a function of its strength and specificity - number of specific conditions.
5. All classifiers are set to not active.
6. New environmental messages are added to list and process is repeated starting at 1.

5.8 Genetic Programming

Q.18 Explain how genetic algorithms are different from evolutionary programming.

Ans. : 1. Genetic algorithms a coded form of the function values i.e. parameter set, rather than with the actual values themselves. For example, if we want to find the minimum of the function $f(x) = x^3 + x^2 + 5$, the GA would not deal directly with x or y values, but with strings that encode these values. For this case, strings representing the binary x values should be used.

2. Genetic algorithms use a set or population of points to conduct a search, not just a single point on the problem space. This gives GAs the power to search noisy spaces littered with local optimum points. Instead of relying on a single point to search through the space, the GAs looks at many different areas of the problem space at once, and uses all of this information to guide it.

3. Genetic algorithms use only payoff information to guide themselves through the problem space. Many search techniques need a variety of information to guide themselves. Hill climbing methods require derivatives, for example. The only information a GA needs is some measure of fitness about a point in the space. Once the GA knows the current measure of "goodness" about a point, it can use this to continue searching for the optimum.

4. GAs are probabilistic in nature, not deterministic. This is a direct result of the randomization techniques used by GAs.

5. GA is inherently parallel. Here lies one of the most powerful features of genetic algorithms. GA's, by their nature, are very parallel, dealing with a large number of points simultaneously.

Parameters	Genetic Algorithms	Traditional Methods
Work with	Coding of parameter set	Parameters directly

Use information	Payoff i.e. objective function	Payoff plus derivatives etc.
Rules	Probabilistic	Fully deterministic
Search	A population of points	A population of points a single point

Q.19 What is difference between evolutionary strategy and evolutionary programming ?

Ans. : • An evolution strategy (ES) is an optimization technique based on ideas of evolution. It belongs to the general class of evolutionary computation or artificial evolution methodologies.

• Evolution strategies use natural problem-dependent representations and primarily mutation and selection, as search operators.

• In common with evolutionary algorithms, the operators are applied in a loop. An iteration of the loop is called a generation. The sequence of generations is continued until a termination criterion is met.

• Evolutionary programming is one of the four major evolutionary algorithm paradigms. It is similar to genetic programming, but the structure of the program to be optimized is fixed, while its numerical parameters are allowed to evolve.

• Genetic Algorithms (GA) and Evolutionary Programming (EP) are two well-known optimization methods that belong to the class of Evolutionary Algorithms (EA).

• Evolutionary algorithms are ubiquitous nowadays, having been success-fully applied to numerous problems from different domains, including optimization, automatic programming, machine learning, operations research, bioinformatics, and social systems. Fig. Q.19.1 shows flow chart of an evolutionary algorithm.

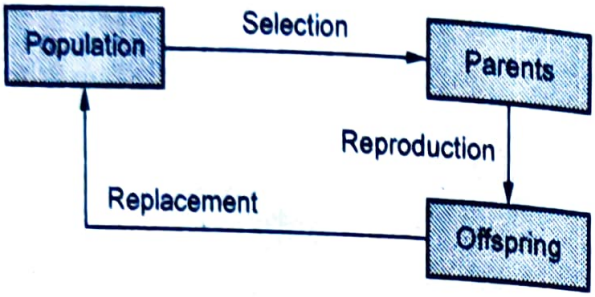


Fig. Q.19.1 Flow chart of an evolutionary algorithm

- A population of candidate solutions is initialized. New solutions are created by applying reproduction operators. The fitness of the resulting solutions are evaluated and suitable selection strategy is then applied to determine which solutions will be maintained into the next generation.

- The basic evolutionary programming method involves the following steps :

  1. Choose an initial population. The number of solutions in a population is highly relevant to the speed of optimization, but no definite answers are available as to how many solutions are appropriate (other than>1) and how many solutions are just wasteful.
  2. New offspring's are created by mutation. Each offspring solution is assessed by computing its fitness.

- When comparing evolutionary programming to genetic algorithm, one can identify the following differences:

  1. GA is implemented by having arrays of bits or characters to represent the chromosomes. In evolutionary programming, there are no such restrictions for the representation. In most cases the representation follows from the problem.
  2. Evolutionary programming typically uses an adaptive mutation operator in which the severity of mutations is often reduced as the global optimum is approached while GA's use a pre-fixed mutation operator. Among the schemes to adapt the mutation step size, the most widely studied being the "meta-evolutionary" technique in which the variance of the mutation distribution is subject to mutation by a fixed

variance mutation operator that evolves along with the solution.

3. The main difference between GA and EP is that GA uses both crossover and mutation, with crossover as the primary search operator, while EP uses only mutation without crossover.

Q.20 What is genetic programming ?

Ans. : • Genetic Programming (GP) is a method to evolve computer programs.

• Genetic programming is a model of programming which uses the ideas of biological evolution to handle a complex problem.

• Genetic programming can be viewed as an extension of the genetic algorithm, a model for testing and selecting the best choice among a set of results, each represented by a string.

• Genetic programming is a recent development in the area of evolutionary computation. It was greatly stimulated in the 1990s by **John Koza.**

• According to Koza, genetic programming searches the space of possible computer programs for a program that is highly fit for solving the problem at hand.

• A parse tree is a good representation of a computer program for Genetic Programming.

5.9 Applications of Genetic Algorithm

Q.21 Mention the application area of genetic algorithms.

Ans. : • Genetic algorithms are global optimization techniques that utilize concurrent search from multiple-points rather than from a single-point. It is independent of the problem complexity. The main necessity of the GA is to specify the objective function and to place finite bounds on the parameters. GA is widely used for robust power system stabilization.

- Optimization using GA techniques are widely applied in many real world problems such as image processing, pattern recognition, classifiers, machine learning.

Traveling Salespersons Problem

- Suppose a salesperson has five cities to visit and then must return home.
- The goal of the problem is to find the shortest path for the salesperson to travel.

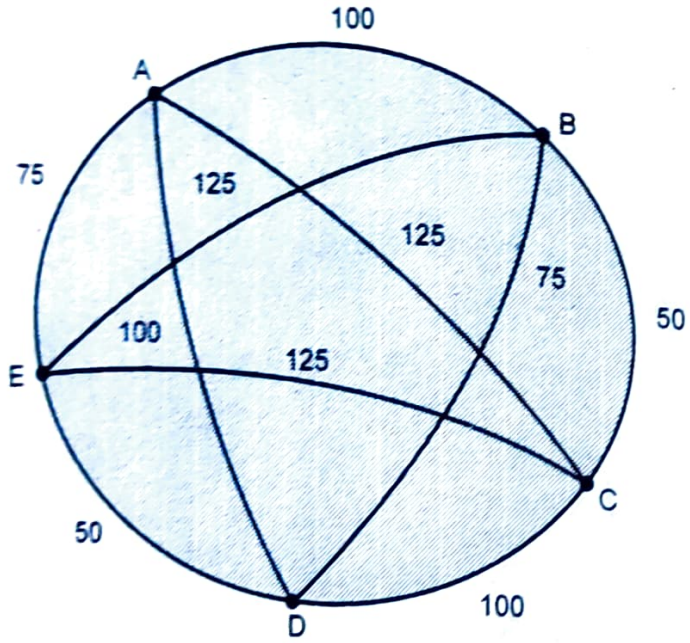


Fig. Q.21.1

- The traveling salesman must visit every city in his territory exactly once (possibly then return to the starting point). Given the cost of travel between all cities, how should he plan his itinerary for minimum total cost of the entire tour ?

Representation :

- Representation is an ordered list of city numbers known as an *order-based* GA.

1) Mumbai	3) Surat	5) Pune	7) Nagpur
2) Calcutta	4) Delhi	6) Bangalore	8) Chennai

CityList1 (3 5 7 2 1 6 4 8)
CityList2 (2 5 7 6 8 1 3 4)

- Crossover combines inversion and recombination :

Parent1 (3 5 7 2 1 6 4 8)
Parent2 (2 5 7 6 8 1 3 4)
Child (5 8 7 2 1 6 3 4)

This operator is called the *Order1* crossover.

- Mutation involves reordering of the list :

```
                *       *
Before :  (5 8 7 2 1 6 3 4)
After :   (5 8 6 2 1 7 3 4)
```

Traveling Salesperson problem with 9 city

- Now, the problem is how to crossover ?

P1 = (192465783)

P2 = (459187623)

- First of all, select two cut point, indicate by a "|", which are randomly inserted into the same location of each parent.

P1 = (192 | 4657 | 83)

P2 = (459 | 1876 | 23)

Two children C1 and C2 are produced in the following way.

First, the segments between cut points are copied into the offspring :

C1 = (XXX | 4657 | XX)

C2 = (XXX | 1876 | XX)

Next, starting from the second cut point of one parent, the cities from the other parent are copied in the same order, omitting cities already present. When the end of the string is reached, continue from the beginning.

Thus, the sequence of cities from P2 (459 | 1876 | 23) is 23 459 1876.

For C1= (XXX | 4657 | XX), once 4657 are removed from the sequence generated by P2, we get the sequence 23918.

Then we just use these numbers to fill in the XXX XX portion in order.

Thus, C1 = (239 | 4657 | 18)

Q.22 Discuss the applications of genetic algorithm : match word finding ?

Ans. : Problem is to 'guess' a word.

- Difficult to solve with a Brute force approach. Assuming case sensitivity, to investigate every possible combination of letters in a seven letter word would require 1,028,071,702,528 attempts. Assume trying to guess the word 'genetic'. Also assign fitness based on number of correct letters in the correct place.
- **Step one :** Select an encoding schema (use characters)
- **Step two :** Initialize the population

1. kdjirid 2. ginddcc 3. nmugjyb
4. zezezez 5. uhjklyt 6. wojikli
7. kladonn 8. flortik

Step three : Evaluate the population

Individual	Genotype	Fitness
1	kdjirid	1
2	ginddcc	3
3	nmugjyb	0
4	zezezez	2
5	uhjklyt	0
6	wojikli	0
7	kladonn	0
8	flortik	2

Average fitness = 1

Step four : Select breeding population and selection based on fitness, so breeding population is,

Individual	Genotype	Fitness
2	ginddcc	3
2	ginddcc	3
2	ginddcc	3
4	zezezez	2
4	zezezez	2
8	flortik	2
8	flortik	2
1	kdjirid	1

Step five : Create new population

Here crossover comes into play and no mutation used here to keep things simple. For example : cross individual 2 with individual 4.

individual 2 genotype is : ginddcc

individual 4 genotype is : zezezez

crossing over produces : genedec

- New population is :

Individual	Genotype	Fitness
1	genedec	5
2	glnrdic	4
3	fdoitik	2
4	zdziziz	1
5	gdnidic	4
6	feoetek	3
7	kijdrcd	0
8	zlzrez	0

Average fitness = 2.375

5.10 Convergence of GA

Q.23 Describe convergence analysis in genetic algorithm.

Ans. : • The convergence of a genetic algorithm arises when the genes of some high rated individuals quickly attain to dominate the population, constraining it to converge to a local optimum. In this case, the genetic operators cannot produce any more descendents better that the parents; the algorithm ability to continue the search for better solutions is therefore substantially reduced.

- The advantages of genetic algorithms first become apparent when a population of strings is observed.
- Let f be the function $x \rightarrow x^2$ which is to be maximized, in the interval [0, 1]. A population of N numbers in the interval [0, 1] is generated in 10-bit fixed-point coding.
- The function f is evaluated for each of the numbers $x_1, x_2, \ldots, x_N$, and the strings are then listed in descending order of their function values. Two strings from this list are always selected to generate a new member of a new population, whereby the probability of selection decreases monotonically in accordance with the ascending position in the sorted list.
- The computed list contains N strings which, for a sufficiently large N, should look like this :

1 0.1 **********

1 0.1 **********

...

...

1 0.0 **********

- The first positions in the list are occupied by strings in which the first bit after the point is a 1. The last positions are occupied by strings in which the first bit after the decimal point is a 0. The asterisk stands for any bit value from 0 to 1 and the zero in front of the point does not need to be coded in the strings.
- The upper strings are more likely to be selected for a recombination, so that the offspring is more likely to contain a 1

the first bit than a 0. The new population is evaluated and a w fitness list is drawn up.

so the strings with a 0 in the first bit are placed at the end of e list and are less likely to be selected than the strings which gin with a 1. After several generations no more strings with a 0 the first bit after the decimal point are contained in the pulation.

e same process is repeated for the strings with a zero in the cond bit. They are also pushed towards extinction. Gradually e whole population converges to the optimal string 1111111111.

ith this quadratic function the search operation is carried out ithin a very well-ordered framework. New points are defined at ch crossover, but steps in the direction x = 1 are more likely an steps in the opposite direction. The step length is also educed in each reproduction step in which the 0 bits are liminated from a position.

ohn Holland suggested the notion of schemata for the onvergence analysis of genetic algorithms. Schemata are bit atterns which function as representatives of a set of binary trings. We already used such bit patterns in the example above : he bit patterns can contain each of the three symbols 0, 1 or *. The schema **00**, for example, is a representative of all strings f length 6 with two zeros in the central positions, such as : 00000, 110011, 010010, etc.

Some authors have argued in favor of the building block ypothesis to explain why GAs do well in some circumstances. According to this hypothesis a GA finds building blocks which are then combined through the generations in order to reach the optimal solution. But the correlations between the optimization parameters sometimes preclude altogether the formation of these building blocks.

Fig. Q.23.1 shows genetic algorithm cycle. Building blocks are combined together due to combined action of generic operators to

form bigger and better building blocks and finally converge to the optimal solution.

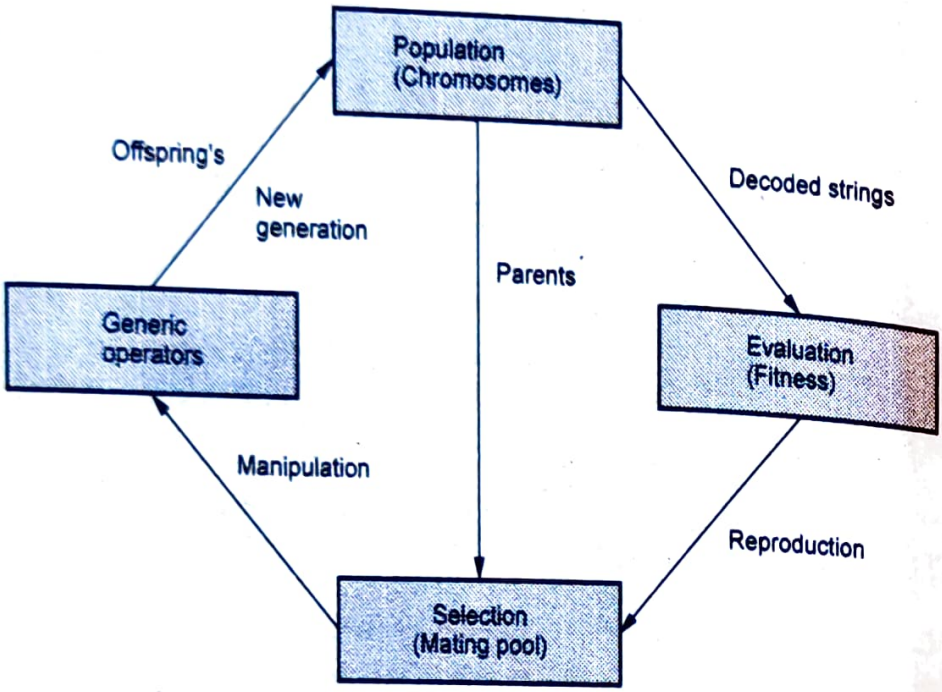


Fig. Q.23.1 Genetic algorithm cycle

Q.24 Explain how genetic algorithm differs from conventional optimization algorithms.

Ans. : • Evolutionary Algorithms (EAs) are a set of probabilistic optimization algorithms based on an analogy between natural biological systems and engineered systems. Genetic algorithms differ from conventional optimization and search procedures in several fundamental ways. It can be summarized as follows :

1. Genetic algorithms work with a coding of solution set, not the solutions themselves.
2. Genetic algorithms search from a population of solutions, not a single solution.
3. Genetic algorithms use payoff information (fitness function), not derivatives or other auxiliary knowledge.
4. Genetic algorithms use probabilistic transition rules, not deterministic rules.
5. GA is robust and has been proven theoretically and empirically to be able to efficiently search complex solution spaces.

Is it advisable to apply genetic algorithm for all kinds of optimization problems ? Justify.

• Optimization techniques are of two types : *deterministic* and *stochastic.*

A common characteristic of **deterministic search techniques** is that they are basically borrowed from deterministic optimization techniques. The deterministic objective function value required in the technique is now replaced with an estimate obtained from simulation. By having a reasonably accurate estimate, one hopes that the technique will perform well. It includes heuristic search, complete enumeration and random search techniques.

Stochastic approach uses random variable.

Here is the situation of genetic algorithms among other well-known search procedures :

For a search space with only a small number of possible solutions, all the solutions can be examined in a reasonable amount of time and the optimal one found. This exhaustive search, however, quickly becomes impractical as the search space grows in size.

Traditional search algorithms randomly sample (e.g., random walk) or heuristically sample (e.g., gradient descent) the search space one solution at a time in the hopes of finding the optimal solution.

• The key aspect distinguishing an evolutionary search algorithm from such traditional algorithms is that it is population-based. Through the adaptation of successive generations of a large number of individuals, an evolutionary algorithm performs an efficient directed search.

• Evolutionary search is generally better than random search and is not susceptible to the hill climbing behaviors of gradient-based search.

- Fig. Q.25.1 shows different classes of search techniques.

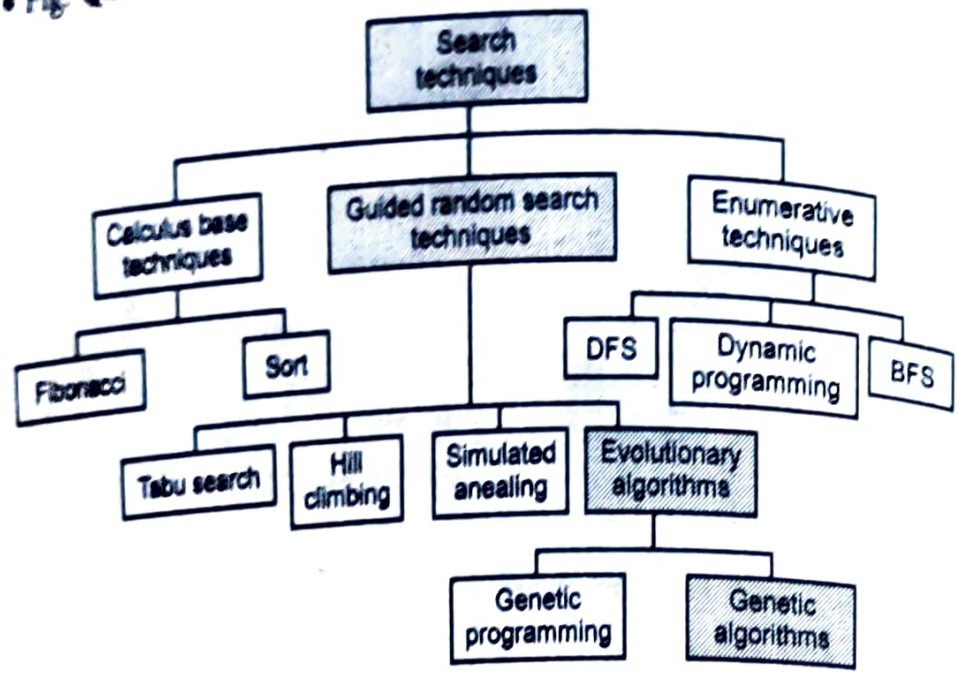


Fig. Q.25.1 Classes of search techniques

END...

UNIT - VI

6 Swarm Intelligence

6.1 : Introduction of Swarm Intelligence

Q.1 What is an agent ? What is a Swarm ? Define swarm intelligence. Give an examples of Swarms in nature.

Ans. : • An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.

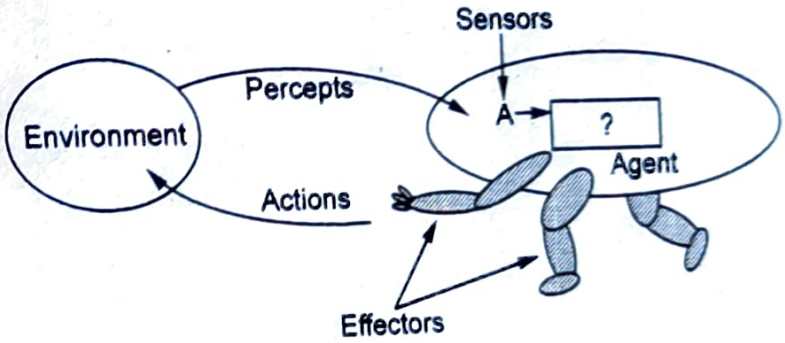


Fig. Q.1.1

- Swarm is a loosely structured collection of interacting agents. Agents is an individuals that belong to a group. They contribute to and benefit from the group. They can recognize, communicate, and/or interact with each other.
- The instinctive perception of swarms is a group of agents in motion, but that does not always have to be the case.
- A swarm is better understood if thought of as agents exhibiting a collective behavior.
- Examples of Swarms in Nature :
  1. Classic Example : Swarm of Bees

2. Can be extended to other similar systems : Ant co (Agents : ants), Flock of birds(Agents : birds), Traffic(Age cars) , Crowd(Agents : humans), Immune system(Agents : and molecules)

- Swarm Intelligence : Any attempt to design algorithms distributed problem-solving devices inspired by the collect behavior of social insect colonies and other animal societies.
- Swarm intelligence (SI) is an artificial intelligence technique bas around the study of collective behavior in decentralize self-organized systems

Q.2 List the advantages of swarm intelligence.

Ans. : • The systems are scalable because the same contro architecture can be applied to a couple of agents or thousands o agents.

- The systems are flexible because agents can be easily added or removed without influencing the structure.
- The systems are robust because agents are simple in design, the reliance on individual agents is small, and failure of a single agents has little impact on the system's performance.
- The systems are able to adapt to new situations easily.

6.2 : Particle Swarm Optimization (PSO) Algorithm

Q.3 Write short note on particle swarm optimization

Ans. : • Particle Swarm Optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.

- PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation.
- Suppose a group of birds is searching food in an area. Only one piece of food is available. Birds do not have any knowledge

about the location of the food. But they know how far the food is from their present location.

- **So what is the best strategy to locate the food ? The best strategy is to follow the bird nearest to the food.**
- A flying bird has a position and a velocity at any time. In search of food, the bird changes his position by adjusting the velocity. The velocity changes based on his past experience and also the feedbacks received from his neighbor.
- This searching process can be artificially simulated for solving non-linear optimization problem. So this is a population based stochastic optimization technique inspired by social behaviour of bird flocking or fish schooling.
- PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms. The system is initialized with a population of random solutions and searches for optima by updating generations.
- In PSO algorithms, the population $P = \{p_1,\ldots,p_n\}$ of the feasible solutions is often called a **swarm.** The feasible solutions $p_1,\ldots,p_n$ are called **particles.**
- However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.
- Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far.
- The fitness value is also stored. This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called lbest. When a particle takes all the population as its topological neighbors, the best value is a global best and is called gbest.

- The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its pbest and lbest locations. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward pbest and lbest locations.
- In past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods.
- **Advantages and disadvantages**
  1. The fitness function can be non-differentiable
  2. There is no general convergence theory applicable to practical, multidimensional problems.

Q.4 Compare Particle Swarm Optimization (PSO) with Genetic Algorithms (GA).

Ans. : • Unlike GAs, PSOs do not change the population from generation to generation, but keep the same population, iteratively updating the positions of the members of the population (i.e., particles).

- PSOs have no operators of "mutation", "recombination", and no notion of the "survival of the fittest".
- On the other hand, similarly to GAs, an important element of PSOs is that the members of the population "interact", or "influence" each other.
- GA usually converges towards a local optimum or even arbitrary points rather than the global optimum of the problem while as PSO tries to find the global optima.
- GA is discrete in nature, i.e. it changes the variables into binary 0's and 1's, and therefore it can easily handle discrete problems, and PSO is continuous and hence must be modified in order to handle discrete problems.

Q.5 Write an algorithm of particle swarm optimization.

Algorithm parameters :

A : Population of agents

p_i : Position of agent a_i in the solution space

f : Objective function

v_i : Velocity of agent's a_i

$V(a_i)$: Neighborhood of agent a_i (fixed)

• Particle update rule : p = p + v

With $v = v + c_1^* \text{ rand} * (pBest - p) + c_2^* \text{ rand} * (gBest - p)$

where

p : Particle's position

v : Path direction

c1 : Weight of local information

c2 : Weight of global information

pBest : Best position of the particle

gBest : Best position of the swarm

rand : Random variable

Algorithm :

1. Create a 'population' of agents (particles) uniformly distributed over X.
2. Evaluate each particle's position according to the objective function.
3. If a particle's current position is better than its previous best position, update it.
4. Determine the best particle (according to the particle's previous best positions).
5. Update particles' velocities :

$$v_i^{t+1} = \underbrace{v_i^t}_{\text{inertia}} + \underbrace{c_1 U_1^t (pb_i^t - p_i^t)}_{\text{personal influence}} + \underbrace{c_2 U_2^t (gb^t - p_i^t)}_{\text{social inflence}}$$

6. Move particles to their new positions :

$$p_i^{t+1} = p_i^t + v_i^{t+1}$$

7. Go to step 2 until stopping criteria are satisfied.

Q.6 What is Binary Particle Swarm Optimization (BPSO) ? How does it work ?

Ans. : • In BPSO, each solution in the population is a binary string. Each binary string is of dimension "n" which is evaluated to give parameter values.

- In the binary PSO, each binary string represents a particle. Strings are updated bit-by-bit based on its current value, the value of that bit in the best (fitness) of that particle to date, and the best value of that bit to date of its neighbors.
- For binary strings, neighbors can be selected in one of several ways. Some examples are : (for a neighbourhood of size k). Neighbours are the k binary strings whose Hamming distance is minimized. For equal Hamming distances, the choices are arbitrary.
- In BPSO, bit-by-bit updates are done probabilistically. In other words, for a chosen bit "d" in a chosen string "i" it is changed to a 1 with a probability "P" that is a function of its predisposition to be a 1, the best value of itself to date, and the best value of its neighbors.
- The (1 – P) is the probability of changing to a zero.Once P is determined, a random number "R" is generated. If R < P, then the bit becomes a 1; otherwise it becomes a zero.
- In BPSO, population has a set of particles. Each individual particle represents a binary decision.
- This decision can be represented by either YES/TRUE=1 or NO/FALSE=0.
- All particles represent their positions through binary values which are 0 or 1.Velocity is restricted within the range {0,1}

- The velocity vector equation and position vector equation are defined as :

1. Velocity vector equation :

$$v_i^n(t+1) = \frac{1}{1+e^{-v_i^n(t)}}$$

2. Position vector equation :

$$x_i^n(t+1) = \begin{cases} 1 & \text{if } r < v_i^n \\ 0 & \text{otherwise} \end{cases}$$

where r is the random number selected from a uniform distribution in [0, 1].

Q.7 Explain flow diagram showing the particle swarm optimization algorithm with Pseudo code.

Ans. : Flow diagram : (See on next page)

Pseudo Code :

```
for each particle
{
   Initialize particle
}
Do until maximum iterations or minimum error criteria
{
   For each particle
   {
      Calculate Data fitness value
      If the fitness value is better than pBest
      {
         Set pBest = current fitness value
      }
      If pBest is better than gBest
      {
         Set gBest = pBest
      }
   }
```

```
For each particle
{
Calculate particle Velocity
Use gBest and Velocity to update particle Data
}
```

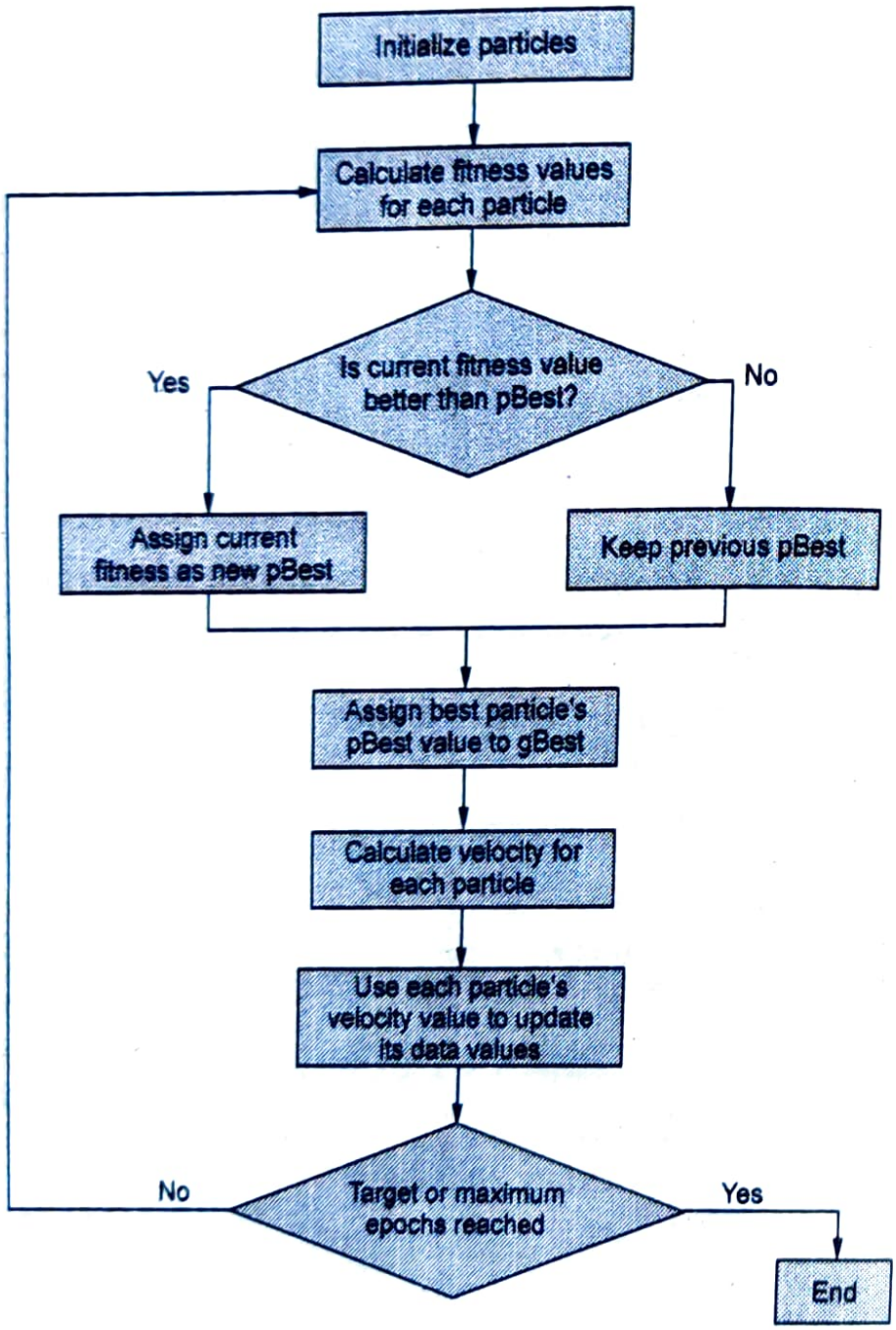


Fig. Q.7.1

Q.8 Discuss topology used in particle swarm optimization.

Ans. : • In PSO, there have been two basic topologies used.

1. Ring Topology (neighborhood of 3)
2. Star Topology (global neighborhood)

Star Topology :

- Fig. Q.8.1 shows star topology.
- gbest model : Each particle is influenced by all the other particles.
- The fastest propagation of information in a population.
- Particles can get stuck easily in local minima.

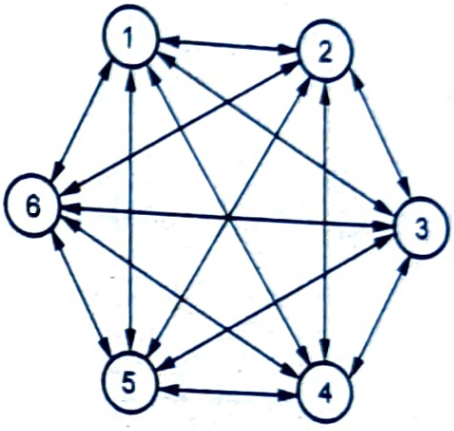


Fig. Q.8.1

- Star neighborhood topology has greater connectivity and interaction between particles compared with other topologies.
- Star neighborhood topology enables fast convergence of the algorithm.

Ring Topology :

- Fig. Q.8.2 shows ring topology.
- lbest model : Each particle is influenced only by particles in its own neighbourhood.
- The propagation of information is the slowest.
- Doesn't get stuck easily in local minima but might increase the computational cost.

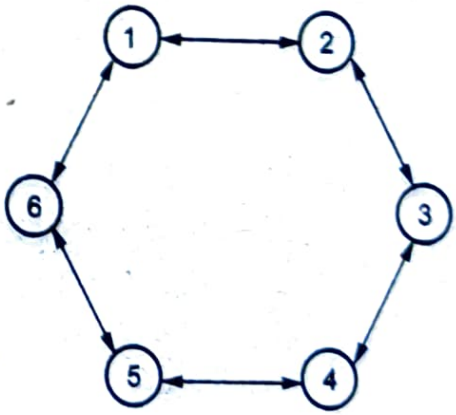


Fig. Q.8.2

- Note that, neighborhood topologies are usually defined by the particle index, and not by the particle location.

Q.9 Explain global best (gbest) and local best (lbest) particle swarm optimization.

Ans. : Global best (gbest) PSO :

- For the global best (gbest) PSO, the neighborhood for each particle is the entire swarm. The social network employed by the gbest PSO reflects the star topology.
- gbest : each particle is influenced by the best found from the entire swarm.
- Velocity update per dimension :

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)]$$

$v_{ij}(0) = 0$ (usually, but can be random) and c_1, c_2 are positive acceleration coefficients.

gbest PSO Algorithm :

Create and initialize an n_x-dimensional swarm, S;

```
repeat
    for each particle i = 1, . . . , S.n_s do
      if f(S·y_i) < f(S·y) then
        S·y_i = S·x_i ;
      end
      if f(S·y_i) < f(S·ŷ) then
          S·ŷ = S·y_i ;
      end
   end
for each particle i = 1, . . . , S·n_s do
update the velocity and then the position;
end
until stopping condition is true ;
```

Local best PSO

- The local best (lbest) PSO, uses a ring social network topology where smaller neighborhoods are defined for each particle.

- The social component reflects information exchanged within the neighborhood of the particle, reflecting local knowledge of the environment.
- With reference to the velocity equation, the social contribution to particle velocity is proportional to the distance between a particle and the best position found by the neighborhood of particles.
- Due to the larger particle interconnectivity of the gbest PSO, it converges faster than the lbest PSO. However, this faster convergence comes at the cost of less diversity than the lbest PSO.
- The lbest model is less vulnerable to the attraction of local optima but with a slower convergence speed than the gbest model.

lbest PSO algorithm :

Create and initialize and n_x-dimensiional swarm;
  repeat
    for each particle i = 1 ... n_s **do**
      //set the personal best position
      if $f(x_i) < f(y_i)$ **then**
        $y_i = x_i$;
      end
      //set the neighborhood best position
      if $f(y_i) < f(\hat{y}_i)$ then
        $\hat{y} = y_i$;
    end
  end
  for each particle i = 1,...n_s do
    update the velocity
    update the position
  end
until stopping condition is true;

Q.10 Why used PSO ?

Ans. :

- With a population of candidate solutions, a PSO algorithm can maintain useful information about characteristics of the environment.
- PSO, as characterized by its fast convergence behavior, has an in-built ability to adapt to a changing environment.
- Some early works on PSO have shown that PSO is effective for locating and tracking optima in both static and dynamic environments.

Q.11 Explain difference between gbest and lbest PSO.

gbest PSO	lbest PSO
Global best particle swarm optimization uses a "star" neighborhood topology where all the particles communicate with each other.	Local best particle swarm optimization can be used with different neighborhood topologies with the "ring" topology being the most common.
gbest PSO converges faster compared with local best particle swarm optimization.	lbest PSO converges is slower compared with global best particle swarm optimization.
For the global best PSO, the neighborhood for each particle is the entire swarm.	In lbest PSO, smaller neighborhoods are defined for each particle.
The gbest PSO performs best for unimodal problems.	In lbest, ring structure to provide better performance in terms of the quality of solutions found for multi-modal problems than the star structure.

6.3 : Ant Colony Optimization

Q.12 Write short note on ant colony optimization.

Ans. : • The way ants find their food in shortest path is interesting.

- Ants secrete pheromones to remember their path. These pheromones evaporate with time.
- Whenever an ant finds food , it marks its return journey with pheromones. Pheromones evaporate faster on longer paths. Shorter paths serve as the way to food for most of the other ants.

Fig. Q.12.1

- The shorter path will be reinforced by the pheromones further. Finally , the ants arrive at the shortest path.
- Ant Colony Optimization (ACO) is a paradigm for designing metaheuristic algorithms for combinatorial optimization problems.
- Metaheuristic algorithms are algorithms which, in order to escape from local optima, drive some basic heuristic : either a constructive heuristic starting from a null solution and adding elements to build a good complete one, or a local search heuristic starting from a complete solution and iteratively modifying some of its elements in order to achieve a better one.

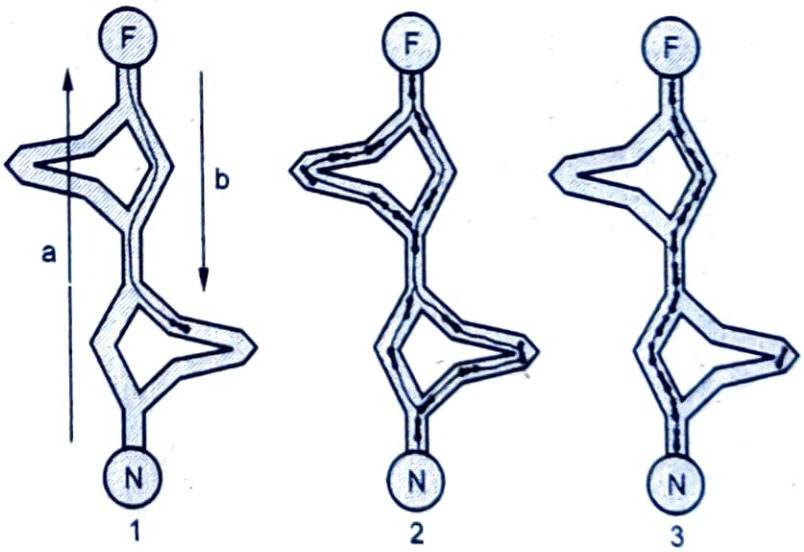


Fig. Q.12.2

- ACO can be used to solve hard problems like TSP, Quadratic Assignment Problem (QAP).
- An ant will move from node i to node j with probability :

$$p_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}$$

where

$\tau_{i,j}$ is the amount of pheromone on edge i, j.

α is a parameter to control the influence of $\tau_{i,j}$.

$\eta_{i,j}$ is the desirability of edge i, j (typically $1/d_{i,j}$)

β is a parameter to control the influence of $\eta_{i,j}$.

- Amount of pheromone is updated according to the equation :

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \Delta\tau_{i,j}$$

where

$\tau_{i,j}$ is the amount of pheromone on a given edge i, j.

ρ is the rate of pheromone evaporation.

$\Delta\tau_{i,j}$ is the amount of pheromone deposited, typically given by,

$$\Delta\tau_{i,j}^{k} = \begin{cases} 1/L_k & \text{if ant k travels on edge i,j} \\ 0 & \text{otherwise} \end{cases}$$

- Where L_k is the cost of the k^{th} ant's tour (typically length).

Q.13 Explain various application of ACO and PSO.

Ans. : 1. Open shop scheduling :

- ACO can be combined with tree search methods with the aim to improve methods for solving combinatorial optimization problems.
- In combination of ACO and beam search is described. It has been applied to open shop scheduling, which is a NP-hard scheduling problem.

- The obtained results compared with two best approaches currently available Standard- show that the Beam-ACO performs better.

2. PSO algorithm in signal detection and blind extraction.

- In signal processing there are among others two important problems, namely multi-user detection and blind extraction of sources.
- The optimal multi-user detection is a NP-complete problem. Therefore the research effort has been focused on the development of suboptimum techniques with the aim to achieve a trade-off between complexity and performance.
- Binary PSO algorithm was used and reached better results than genetic algorithm. Similar results were achieved in blind source separation, which is an important issue in many science and engineering scenarios as well.

3. PSO for Solving Travelling Salesman Problem (TSP)

- TSP is a well-known NP-hard problem. Modification of PSO using fuzzy matrices to represent the position and velocity of the particles in PSO was developed.
- The algorithm can be used for resolving common routing problems and other combinatorial optimization problems.
- Discovering clusters in spatial data. In this application, each agent represents a simple task and the success of the method depends on the cooperative work of the agents.
- The algorithm combines a smart exploratory strategy based on the movements of a flock of birds with a shared nearest-neighbour clustering algorithm to discover clusters in parallel.
- Birds are used as agents with an exploring behaviour foraging for clusters. This strategy can be used as a data reduction technique to perform approximate clustering efficiently.

4. Discovering clusters in biomedical signals

- In this application, individual parts of the signal are characterized by real valued features and the resulting set of samples is partitioned into groups using a clustering method based on PSO algorithm.
- The signal is segmented according to the resulting division into the corresponding groups, where each group represents certain type of segments, thus a certain cluster. The clustering method is applied on real data extracted from electrooculographic signals.

5. Combination of ants and cellular automata

- Application to clustering problem in data mining.
- Ant Sleeping Model combines advantages of the classical cellular automata model and the swarm intelligence.
- The ants have two states: sleeping state and active state.
- The ant's state is controlled by a function of the ant's fitness to the environment it locates and a probability for the ants becoming active.
- The state of an ant is determined only by its local information. By moving dynamically, the ants form different subgroups adaptively.
- The algorithm was applied to clustering problem in data mining. Results show that the algorithm is significantly better than other clustering methods in terms of both speed and quality. It is adaptive, robust and efficient, achieving high autonomy, simplicity and efficiency.

Q.14 List advantages and disadvantages of ant colony optimization.

Ans. :

Advantages :

1. Inherent parallelism
2. Positive feedback accounts for rapid discovery of good solutions

3. Efficient for Traveling Salesman Problem and similar problems
4. It can be used in dynamic applications.

Disadvantages :

1. Theoretical analysis is difficult
2. Sequences of random decisions
3. Probability distribution changes by iteration
4. Research is experimental rather than theoretical
5. Time to convergence uncertain.

END...