easy - solutions

Soft Computing and Optimization Algorithms

(Elective - III)

Semester VIII - Computer Engineering

(Savitribai Phule Pune University)

INDEX

8EP2A5

SYLLABUS

Soft Computing and Optimization Algorithms

Course Contents	
Unit I : Introduction	Introduction, soft computing vs. hard computing, various types of soft computing techniques and applications of soft computing. Basic tools of soft computing - Fuzzy logic, neural network, evolutionary computing. Introduction : Neural networks, application scope of neural networks, fuzzy logic, genetic algorithm and hybrid systems.
Unit II : Fuzzy Sets and Logic	Basic concepts of fuzzy logic, Fuzzy sets and Crisp sets, Fuzzy set theory and operations, Properties of fuzzy sets, Fuzzy and Crisp relations, Fuzzy to Crisp conversion. Membership functions, interference in fuzzy logic, fuzzy if-then rules, Fuzzy implications and Fuzzy algorithms, Fuzzifications and Defuzzifications.
Unit III : Fuzzy Systems	Fuzzy Controller, Fuzzy rule base and approximate reasoning: truth values and tables in fuzzy logic, fuzzy propositions, formation of rules, decomposition of compound rules, aggregation of fuzzy rules, fuzzy reasoning, fuzzy inference system, fuzzy expert systems.
Unit IV : Evolutionary Computing	Basic Evolutionary Processes, EV : A Simple Evolutionary System, Evolutionary Systems as Problem Solvers, A Historical Perspective, Canonical Evolutionary Algorithms - Evolutionary Programming, Evolution Strategies, A Unified View of Simple EAs- A Common Framework, Population Size.
Unit V : Genetic Algorithm	Basic concepts, working principle, procedures of GA, flow chart of GA, Genetic representations, (encoding) Initialization and selection, Genetic operators, Mutation, Generational Cycle, Traditional algorithm vs genetic algorithm, simple GA, general genetic algorithm, schema theorem, Classification of genetic algorithm, Holland classifier systems, genetic programming, applications of genetic algorithm, Convergence of GA. Applications and advances in GA, Differences and similarities between GA and other traditional method, applications.
Unit VI : Swarm Intelligence	Swarm intelligence, Particle Swarm Optimization (PSO) Algorithm-Formulations, Pseudo-code, parameters, premature convergence, topology, biases, Real valued and binary PSO, Ant colony optimization (ACO) - Formulations, Pseudo-code, Applications of PSO and ACO.

□□□

Soft Computing & Optimization Algorithm

Chapter 1 : Introduction

Q. 1 Define soft computing.

Ans. :

Soft computing consists of distinct concepts and techniques that help in solving complex real-world problems that are difficult to solve using conventional methodologies. Soft computing is a collection of all the techniques that help us to construct computationally intelligent systems.

Q. 2 Differentiate between soft computing and hard computing.

Ans. :

Differences between hard computing and soft computing

Sr. No.	Hard computing	Soft computing
1.	Hard computing is a conventional type of computing that requires a precisely stated analytic model.	Soft computing techniques are imprecision, approximation and uncertainty tolerant.
2.	Hard computing requires programs to be written.	Soft computing techniques are model free. They can evolve their own models and programs.
3.	Hard computing is deterministic and uses two-valued logic.	Soft computing is stochastic and uses multi-valued logic such as fuzzy logic.
4.	Hard computing needs exact data to solve a particular problem.	Soft computing can deal with incomplete, uncertain and noisy data.
5.	Hard computing techniques perform sequential computation.	Soft computing allows parallel computations. E.g. Neural networks.
6.	The solution or output of hard computing is precise.	Soft computing can generate approximate output or solution.
7.	Hard computing is based on crisp logic, binary logic and numerical analysis.	Soft computing is based on neural networks, fuzzy logic, and evolutionary computations etc.
8.	Hard computing techniques are not fault tolerant. The reason is conventional programs and algorithms are built in such a way that errors have serious consequences, unless enough redundancy is added into the system.	Soft computing techniques are fault tolerant due to their redundancy, adaptability and reduced precision characteristics.

Q. 3 What are the constituents (or types) of Soft Computing techniques ? Explain each in brief.

Ans. :

Soft Computing is the fusion of different techniques that were designed to model and enable solutions to complex real world problems, which are not modeled or too difficult to model, mathematically. Soft computing consist several computing paradigms mainly are :

1. Neural Network
2. Fuzzy Logic
3. Evolutionary Algorithms such as Genetic algorithm

Every paradigm of soft computing mentioned above has its own strength. In order to build a computationally intelligent system, we may integrate multiple techniques or methodologies to take advantage of the strengths of each of them. Such systems are called **Hybrid soft computing systems.**

Table 1.1 summarizes the soft computing methodologies and their strengths.

Table 1.1 : Soft computing constituents and their strengths

Sr. No.	Methodology	Strengths
1.	Neural Networks	Has capability of learning and adaptation.

Sr. No.	Methodology	Strengths
2.	Fuzzy set theory	Handles uncertainty and incorporates human-like reasoning into the system.
3.	Evolutionary algorithms	Has capability of finding optimum solution to a problem.

The seamless integration of these methodologies forms the base of soft computing.

Neural networks have the capability of recognizing patterns and adapting themselves to cope with changing environments.

The evolutionary algorithms such as Genetic Algorithms are search and optimization techniques based on biological evolution that help us to optimize certain parameters in a given problem.

Fuzzy logic incorporates human knowledge and performs inference and decision making.

Q. 4 State application of soft computing.

Ans. : Applications of soft computing

Soft computing techniques are used almost in every area. Some of the applications of soft computing are as follows,

1. Handwriting recognition system using neural network
2. Image processing and data compression
3. Automotive system and manufacturing
4. Robotics
5. Medical image processing
6. Processing natural languages

Q. 5 What are the advantages of neural networks.

Ans. : Advantages of neural networks

1. Neural networks provide human like artificial intelligence.
2. A neural network learns and does not need to be reprogrammed.
3. A neural network can do a task that a linear program cannot do.
4. Parallel organization of neural networks permits solutions to problems where multiple constraints must be satisfied simultaneously.
5. Because of its parallel nature, when an element of the neural network fails, it can continue without any problem.

Q. 6 Give the application scope of neural networks.

Ans. : Application scope of neural networks

Neural networks have been successfully applied to a broad spectrum of data-intensive applications. Few of them are listed below.

1. Forecasting
2. Image compression
3. Industrial process control
4. Optical Character Recognition
5. Customer Relationship Management
6. Medical science

Q. 7 Give advantages of fuzzy logic controllers.

Ans. : Advantages of fuzzy logic controllers

1. Simplicity and flexibility
2. Can handle problems with imprecise and incomplete data
3. Can model nonlinear functions of arbitrary complexity
4. Cheaper to develop,
5. Cover a wider range of operating conditions, more readily customizable in natural language terms.

Q. 8 What are the applications of fuzzy logic ?

Ans. : Applications of fuzzy logic

1. Fuzzy logic can be used in applications where human like decision making with an ability to generate precise solutions from certain or approximate information is required.
2. Fuzzy logic has been extensively used in design of controllers for home appliances such as washing machine, vacuum cleaner, air conditioner etc.
3. Fuzzy logic can also be used for other applications such as facial pattern recognition, anti-skid braking systems, transmission systems, control of subway systems and unmanned helicopters.
4. Another application area of fuzzy logic is development of knowledge-based systems for multi objective optimization of power systems, weather forecasting systems, models for new product pricing or project risk assessment, medical diagnosis and treatment plans, and stock trading.
5. Fuzzy logic has been successfully used in numerous fields such as control systems engineering, image processing, power engineering, industrial automation, robotics, consumer electronics and optimization.

Q. 9 Write short note on : Evolutionary computing.

Ans. :

Evolutionary computing refers to the class of algorithms that are inspired by the biological evolution process. They are heuristic-based approach to solve problems that cannot be easily solved in polynomial time, such as classically NP-Hard problems.

In evolutionary computation, an initial set of candidate solutions is generated and iteratively updated. Each new generation is produced by stochastically removing less desired solutions, and introducing small random changes. Evolutionary computation techniques can produce highly optimized solutions in a wide range of problem settings

Many variations of evolutionary computation exist. They are :

There are three basic types of evolutionary algorithms, namely :

1. Genetic algorithms
2. Evolutionary programming
3. Evolutionary strategies

Unlike traditional optimization techniques, evolutionary algorithms depend on random sampling. An evolutionary algorithm has a population of candidate solutions, unlike classical methods, which try to maintain a single best solution.

Usually evolutionary algorithms contain four overall steps : **initialization, selection, genetic operators and termination.**

Evolutionary algorithms make use of concepts in biology such as selection, reproduction and mutation.

Q. 10 Explain the basics of genetic algorithm along with its applications.

Ans. :

Genetic Algorithms are ***adaptive heuristic search*** algorithms based on the evolutionary ideas of natural selection and genetics. GAs are often used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve. A GA can efficiently explore a large space of candidate designs and find optimum solutions.

GAs is a subset of an **Evolutionary Computation**. GAs were developed by John Holland and his students and colleagues at the University of Michigan. In GAs, we select the **initial pool or a population of possible solutions** to the given problem. These solutions then undergo various GA operations like recombination and mutation which in turn produce new children.

The process is repeated over various generations. Each individual or candidate solution is assigned a fitness 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". Thus GA keeps "evolving" better individuals or solutions over generations, till it reaches a stopping criterion. Genetic Algorithms are sufficiently randomized in nature, but they perform much better than random local search.

Applications of Genetic Algorithms

1. **Automotive Design** : Genetic algorithms can be used to design composite materials and aerodynamic shapes for race cars to provide faster, lighter, more fuel efficient and safer vehicles for all the things we use vehicles for.
2. **Engineering Design** : GA are most commonly used to optimize the structural and operational design of buildings, factories, machines, etc. GA s are used for optimizing the design of robot gripping arms, satellite booms, building trusses turbines, , flywheels or any other computer-aided engineering design application.
3. **Robotics** : GAs has found applications that span the range of architectures for intelligent robotics. GAs can be used to design the entirely new types of robots that can perform multiple tasks and have more general application.

Q. 11 What are the advantages of genetic algorithms?

Ans. : Advantages of Genetic Algorithms

1. GAs are easy to understand since they do not demand the knowledge of complex mathematics.
2. Does not require any derivative information
3. Good for noisy environment.
4. Easy to discover global optimum.
5. They can solve multimodal, non differentiable, non continuous or even NP-complete problems.

Q. 12 What are hybrid systems? Explain different types of hybrid systems.

Ans. :

Hybrid systems are those for which more than one soft computing technique is integrated to solve a real-world problem.

Types of Hybrid Systems

1. Sequential Hybrid Systems

In sequential hybrid systems, all the technologies are used in a pipe-line fashion. The output of one technology becomes the input to another technology (Fig. 1.1). This kind of hybridization form is one of its weakest, because it does not integrate different technologies into a single unit. An example is a GA pre-processor which obtains the optimal parameters such as initial weights, threshold, learning rate etc. and hands over these parameters to a neural network.

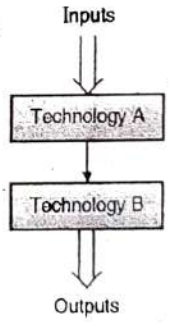


Fig. 1.1 : A sequential hybrid system

2. Auxiliary Hybrid Systems

In this type, a technology treats another technology as a "subroutine" and calls it to process or generate whatever information is needed by it. Fig. 1.2 illustrates the auxiliary hybrid system.

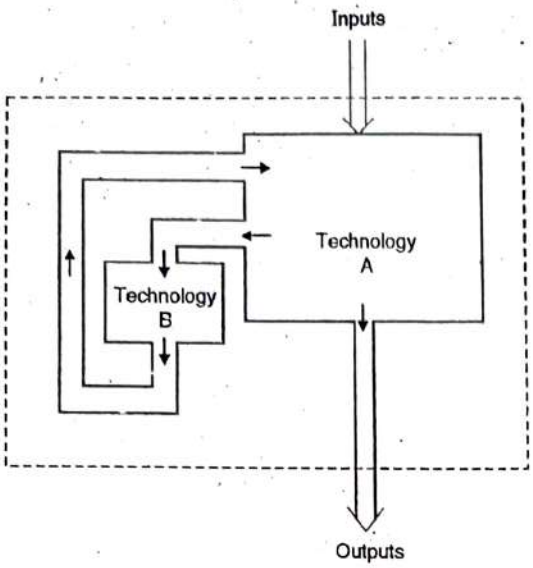


Fig. 1.2 : An auxiliary system

An example is a neuro-genetic system in which an NN employs a GA to optimize its structural parameters, i.e. parameters which define Neural Network's architecture.

3. Embedded Hybrid Systems

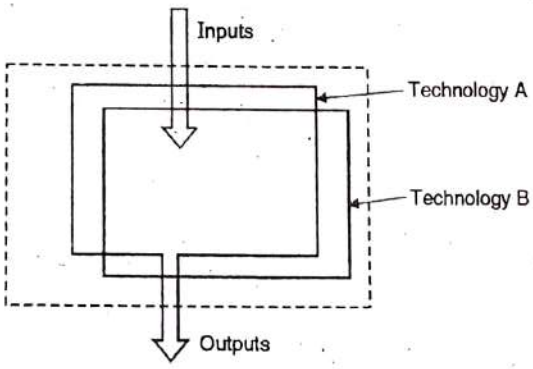


Fig. 1.3 : An embedded system

In embedded hybrid systems, the technologies are integrated in such a manner that they appear to be intertwined. The fusion is so complete that it would appear that no technology can be used without the other for solving the problem. Fig. 1.3 depicts the schema for an embedded hybrid system.

Example of this system is a NN-FL (Neural Network Fuzzy Logic) hybrid system that has NN which receives information, processes it and generates fuzzy outputs as well.

Q. 13 What are the applications of hybrid systems?

Ans. :

Hybrid systems can be used in almost all areas ranging from medical science, engineering, aviation, image processing, business applications etc. Few of them are as follows:

1. Process control

Process control is an important application of any industry for controlling the complex system parameters, which can readily benefit from hybrid soft computing systems. Hybrid artificial intelligent techniques provide more robust and reliable problem solving models than standalone models. Integrating these techniques enhance the overall strengths and lessen weakness thereby helping to solve overall control problem in effective way.

For example, we can use fuzzy logic with genetic algorithm to control the direct torque of induction motor.

2. Medical Image classification

Image classification is extensively used in medical science in which the abnormal images are differentiated from the normal ones. Researchers have proved that hybrid systems such as ANN with the integration of fuzzy or ANN with genetic algorithm give better results than the standalone system.

Q. 13 Give examples of hybrid system.

Ans. :

Examples of the hybrid systems are,

1. A Neuro-fuzzy hybrid system
2. Genetic-Neuro hybrid systems
3. Fuzzy Genetic Hybrid System

Chapter 2 : Fuzzy Sets and Logic

Q. 1 Why fuzzy logic is required?

Ans. :

Most of our traditional tools for formal modelling, reasoning and computing are crisp, deterministic and precise. While designing the system using classical set, we assume that the structures and parameters of the model are definitely known and there are no doubts about their values or their occurrence.

But in real world there exists much fuzzy knowledge; knowledge that is vague, imprecise, uncertain, ambiguous, inexact or probabilistic in nature.

There are two facts;

1. Real situations are very often not crisp and deterministic and they cannot be described precisely.
2. The complete description of a real system often would require more detailed data than a human being could ever recognize simultaneously, process and understand.

Because of these facts, modeling the real system using classical sets often do not reflect the nature of human concepts and thoughts which are abstract, imprecise and ambiguous. The classical (crisp) sets are unable to cope with such unreliable and incomplete information. We want our systems should also be able to cope with unreliable and incomplete information and give expert opinion.

Fuzzy set theory has been introduced to deal with such unreliable, incomplete, vague and imprecise information. Fuzzy set theory is an extension to classical set theory where element have degree of membership. Fuzzy logic uses the whole interval between 0 (false) and 1 (true) to describe human reasoning.

Q. 2 Write a short note on : Crisp sets.

Ans. :

A classical set (or conventional or crisp set) is a set with a crisp boundary.

For example, a classical set A of real numbers greater than 6 can be expressed as

$$A = \{x \mid x > 6\}$$

Where there is a clear, unambiguous boundary '6' such that if x is greater than this number, then x belongs to the set A, otherwise x does not belong to the set.

Although classical sets are suitable for various approximations and have proven to be an important tool for mathematics and computer science, they do not reflect the nature of human concepts and thoughts, which are abstract, imprecise and ambiguous.

For example, mathematically we can express the set of all tall persons as a collection of persons whose height is more than 6 ft.

$$A = \{x \mid x > 6\}$$

Where A = "tall person" and x = "height".

The problem with the classical set is that it would classify a person 6.001 ft. tall as a tall person, but a person 5.999 ft. tall as "not tall". This distinction is intuitively unreasonable.

The flaw comes from the sharp transition between inclusion and exclusion in a set.

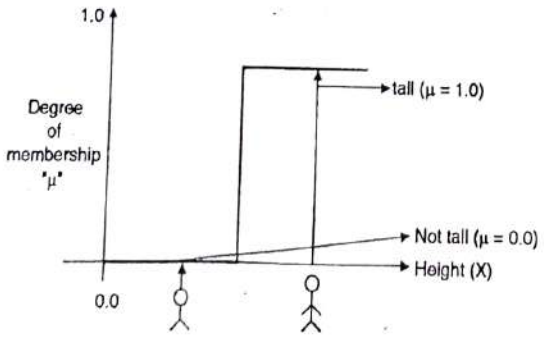


Fig. 2.1 : Sharp edged membership function for TALL

Q. 3 Explain operations on classical / crisp sets.

Ans. : Operations on classical / crisp sets

1. Union

The union of two classical sets A and B is given by the set of all elements which belong to either set A or set B and is denoted by $A \cup B$.

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$$

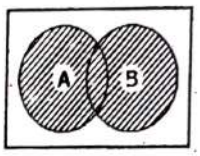


Fig. 2.2 : Union of A and B

2. Intersection

The intersection of two classical sets A and B is represented by the set of all the elements which belong to both A and B and is denoted by $A \cap B$.

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$$

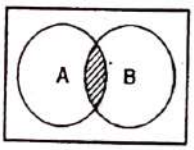


Fig. 2.3 : Intersection of A and B

3. Complement

The complement of set A is defined as the collection of all elements that do not belong to A and is denoted by $\bar{A}$.

$$\bar{A} = \{x \mid x \notin A,\ x \in X\}$$

Where X is the universal set and A is a given set formed from universe X.

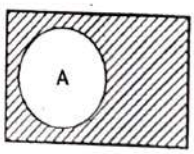


Fig. 2.4 : Complement of set A

4. Difference (Subtraction)

The difference of set A with respect to set B is the collection of all elements in the universe which belong to A but do not belong to B and is denoted by A\B or A – B.

$$A \setminus B \text{ or } A - B = \{x \mid x \in A \text{ and } x \notin B\} = A - (A \cap B)$$

Similarly, the difference of set B w.r.t. set A is given by,

$$B \setminus A \text{ or } B - A = \{x \mid x \in B \text{ and } x \notin A\} = B - (B \cap A)$$

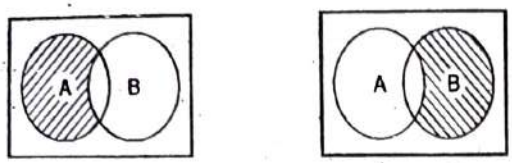


Fig. 2.5 : Difference operation

Q. 4 What are the properties of crisp sets ?

Ans. : Properties of crisp sets

1) **Commutativity**

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

2) **Associativity**

$$A \cup (B \cup C) = (A \cup B) \cup C$$

$$A \cap (B \cap C) = (A \cap B) \cap C$$

3) **Distributivity**

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

4) **Idempotency**

$$A \cup A = A$$

$$A \cap A = A$$

5) **Transitivity**

If $A \subseteq B \subseteq C$, then $A \subseteq C$

6) **Identity**

$$A \cup \phi = A, \quad A \cap \phi = \phi$$

$$A \cup X = X, \quad A \cap X = A$$

7) **Involution**

$$\bar{\bar{A}} = A$$

8) **Law of excluded middle**

$$A \cup \bar{A} = X$$

9) **Law of contradiction**

$$A \cap \bar{A} = \phi$$

10) **De-Morgan's law**

$$\overline{A \cap B} = \bar{A} \cup \bar{B}$$

$$\overline{A \cup B} = \bar{A} \cap \bar{B}$$

Q. 5 Define fuzzy sets.

Ans. :

If X is a collection of objects denoted generally by x, then a fuzzy set $\tilde{A}$ in X is defined as a set of ordered pairs :

$$\tilde{A} = \{ (x, \mu_{\tilde{A}}(x)) \mid x \in X \}$$

Where, $\mu_{\tilde{A}}(x)$ is called the membership function (MF for short) for the fuzzy set $\tilde{A}$.

Q. 6 State the different properties of fuzzy set.

Ans. : Properties of fuzzy sets

Fuzzy sets follow the same properties as crisp set except for the law of excluded middle and law of contradiction.

That is, for fuzzy set $\tilde{A}$

$$\tilde{A} \cup \bar{\tilde{A}} \neq U \quad ; \quad \tilde{A} \cap \bar{\tilde{A}} \neq \phi$$

The following are the properties of fuzzy sets :

1. **Commutativity**

$$\tilde{A} \cup \tilde{B} = \tilde{B} \cup \tilde{A} \quad \text{and}$$

$$\tilde{A} \cap \tilde{B} = \tilde{B} \cap \tilde{A}$$

2. **Assoiciativity**

$$\tilde{A} \cup (\tilde{B} \cup \tilde{C}) = (\tilde{A} \cup \tilde{B}) \cup \tilde{C}$$

$$\tilde{A} \cap (\tilde{B} \cap \tilde{C}) = (\tilde{A} \cap \tilde{B}) \cap \tilde{C}$$

3. **Distributivity**

$$\tilde{A} \cup (\tilde{B} \cap \tilde{C}) = (\tilde{A} \cup \tilde{B}) \cap (\tilde{A} \cup \tilde{C})$$

$$\tilde{A} \cap (\tilde{B} \cup \tilde{C}) = (\tilde{A} \cap \tilde{B}) \cup (\tilde{A} \cap \tilde{C})$$

4. **Identity**

$$\tilde{A} \cup \phi = \tilde{A} \quad ; \quad \tilde{A} \cup U = U$$

$$\tilde{A} \cap \phi = \phi \quad ; \quad \tilde{A} \cap U = \tilde{A}$$

5. **Involution**

$$\bar{\bar{\tilde{A}}} = \tilde{A}$$

6. **Transitivity**

If $\tilde{A} \subset \tilde{B} \subset \tilde{C}$, then $\tilde{A} \subset \tilde{C}$

7. **De Morgan's law**

$$\overline{\tilde{A} \cup \tilde{B}} = \bar{\tilde{A}} \cap \bar{\tilde{B}}$$

$$\overline{\tilde{A} \cap \tilde{B}} = \bar{\tilde{A}} \cup \bar{\tilde{B}}$$

Q. 7 Explain crisp (classical) relations.

Ans. :

An n-ary relation over $M_1, M_2, M_3, \ldots M_n$ is a subset of the Cartesian product $M_1 \times M_2 \times \ldots M_n$. Where n = 2, the relation is a subset of the Cartesian product $M_1 \times M_2$. This is called a binary relation from M_1 to M_2.

Assuming X and Y are two universe and $X \times Y$ is their Cartesian product.

Then $X \times Y$ can be defined as,

$$X \times Y = \{(x, y) \mid x \in X, y \in Y\}$$

Every element in X is related to every element in Y.

The characteristic function f can be defined that gives the strength of the relationship between the each element of X and Y.

$$f_{X \times Y}(x, y) = \begin{cases} 1 & , (x, y) \in X \times Y \\ 0 & , (x, y) \notin X \times Y \end{cases}$$

The relation can be represented in the form of matrix.

An n-dimensional relation matrix represents on n-ary relation.

So, binary relation is represented by 2 dimensional matrices.

Example : Considering the following two universe,

$$X = \{a, b, c\}, \quad Y = \{1, 2, 3\}$$

The Cartesian product - $X \times Y$ is,

$$X \times Y = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3), (c, 1), (c, 2), (c, 3)\}$$

From the above set, we may select a subset R, such that

$$R = \{(a, 1), (b, 2), (b, 3), (c, 1), (c, 3)\}$$

Then R can be represented in matrix form as,

R	1	2	3
a	1	0	0
b	0	1	1
c	1	0	1

The relation between set X and Y can also be represented as coordinate diagram as shown in Fig. 2.6.

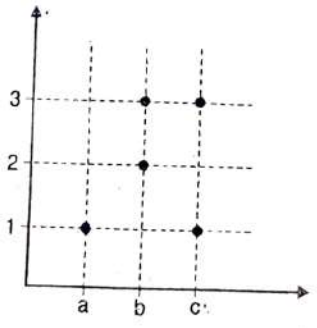


Fig. 2.6 : Co-ordinate diagram of a relation

The relation R can also be expressed by mapping representation as shown in Fig. 2.7.

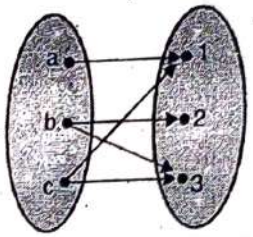


Fig. 2.7 : Mapping representation of a relation

A characteristic function is used to assign values of relationship in the mapping of $X \times Y$ to the binary values and is given by,

$$f_{R(x,y)} = \begin{cases} 1 & , (x, y) \in R \\ 0 & , (x, y) \notin R \end{cases}$$

Q. 8 Explain operations on classical relations.

Ans. :

Considering A and B are two separate relations defined on the Cartesian universe $X \times Y$.

Then the null relation defined as,

$$\phi_A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

And complete relation is defined as,

$$E_A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The following operations can be performed on two relations A and B.

1. **Union**

$$A \cup B \rightarrow f_{A \cup B}(x, y) : f_{A \cup B}(x, y) = \max [f_A(x, y), f_B(x, y)]$$

2. **Intersection**

$$A \cap B \rightarrow f_{A \cap B}(x, y) : f_{A \cap B}(x, y) = \min [f_A(x, y), f_B(x, y)]$$

3. **Complement**

$$\bar{A} \rightarrow f_{\bar{A}}(x, y) : f_{\bar{A}}(x, y) = 1 - f_{\bar{A}}(x, y)$$

4. **Containment**

$$A \subset B \rightarrow f_A(x, y) : f_B(x, y) \leq f_B(x, y)$$

5. **Identity**

$$\phi \rightarrow \phi_A \text{ and } X \rightarrow E_A$$

Q. 9 Explain types of operations Fuzzy Relation.

Ans. :

Fuzzy relations are very important in fuzzy controller because they can describe interaction between variables.

Four types of operations can be performed on fuzzy relation.

1. Intersection

Assuming R and S are binary relations defined on X × Y. The intersection of R and S is defined by,

$\forall (x, y) \in X \times Y : \mu_{R \cap S}(x, y) = \min(\mu_R(x, y), \mu_s(x, y))$

Instead of the minimum, any T - Norm can be used.

2. Union

The union of R and S is defined as,

$\forall (x, y) \in X \times Y : \mu_{R \cup S}(x, y) = \max(\mu_R(x, y), \mu_s(x, y))$.

Instead of maximum, any S – norm can be used.

Given two relations R and S

R =

	y_1	y_2	y_3
x_1	0.3	0.2	0.1
x_2	0.4	0.6	0.1
x_3	0.2	0.3	0.5

S =

	y_1	y_2	y_3
x_1	0.4	0	0.1
x_2	1	0.2	0.8
x_3	0.3	0.2	0.4

Then using max operation,

$R \cup S$ =

0.4	0.2	0.1
1	0.6	0.8
0.3	0.3	0.5

Suppose a simple S - norm

$S(a, b) = a + b - a \cdot b$ is used then,

$R \cup S$ =

0.58	0.2	0.19
1	0.68	0.84
0.44	0.44	0.7

This operation is more optimistic than the max operation. All the membership degrees are at least as high as in the max operation.

Using min operation,

$R \cap S$ =

0.3	0	0.1
0.4	0.2	0.1
0.2	0.2	0.4

Suppose a simple T norm $T(a, b) = \frac{a \cdot b}{a + b - a \cdot b}$ is used then,

$R \cap S$ =

0.20	0	0.1
0.4	0.17	0.10
0.13	0.13	0.28

The above operation is more optimistic than the min operation. All the membership degrees are less than in the min operation.

3. Projection

The projection relation brings a ternary relation back to a binary relation, or a binary relation to a fuzzy set, or a fuzzy set to a single crisp value.

Ex. Consider the relation R as given below.

	y_1	y_2	y_3	y_4
x_1	0.8	1	0.1	0.7
x_2	0	0.8	0	0
x_3	0.9	1	0.7	0.8

Then the projection on X means that

x_1 is assigned the maximum of the first row.

x_2 is assigned the maximum of the second row.

x_3 is assigned the maximum of the third row.

Thus,

$$\text{Proj. R on X} = \frac{1}{x_1} + \frac{0.8}{x_2} + \frac{1}{x_3}$$

Similarly,

$$\text{Proj. R on Y} = \frac{0.9}{y_1} + \frac{1}{y_2} + \frac{0.7}{y_3} + \frac{0.8}{y_4}$$

4. Cylindrical Extension

The projection operation is almost always used in combination with cylindrical extension.

Cylindrical extension is more or less opposite of projection.

It converts fuzzy set to a relation.

e.g. consider a fuzzy set,

A = proj. of R on $X = 1/x_1 + 0.8/x_2 + 1/x_3$.

Its cylindrical extension on the domain X × Y is

ce (A) =

	y_1	y_2	y_3	y_4
x_1	1	1	1	1
x_2	0.8	0.8	0.8	0.8
x_3	1	1	1	1

Considering the fuzzy set,

$$B = \text{proj. of R on X} = \frac{0.9}{y_1} + \frac{0.8}{y_2} + \frac{0.7}{y_3} + \frac{0.8}{y_4}$$

ce (B) =

	y_1	y_2	y_3	y_4
x_1	0.9	0.8	0.7	0.8
x_2	0.9	0.8	0.7	0.8
x_3	0.9	0.8	0.7	0.8

Q. 10 What are the properties of fuzzy relations ?

Ans. :

Assuming R, S and T are fuzzy relations defined on the universe X × Y. Then, the properties of fuzzy relations are stated below :

1. **Commutativity**

$$R \cup S = S \cup R$$

$$R \cap S = S \cap R$$

2. **Associativity**

$$R \cup (S \cup T) = (R \cup S) \cup T$$

$$R \cap (S \cap T) = (R \cap S) \cap T$$

3. **Distributivity**

$$R \cup (S \cap T) = (R \cup S) \cap (R \cup T)$$

$$R \cap (S \cup T) = (R \cap S) \cup (R \cap T)$$

4. **Idempotency**

$$R \cup R = R$$

$$R \cap R = R$$

5. **Identity**

$$R \cup \phi_R = R \ , \quad R \cap \phi_R = \phi_R$$

$$R \cup E_R = E_R \ , \quad R \cap E_R = R$$

Where ϕ_R and E_R are null relation (null matrix) and complete relation (unit matrix of all 1s) respectively.

6. **Involution**

$$\bar{\bar{R}} = R$$

7. **De-Morgan's law**

$$\overline{R \cap S} = \bar{R} \cup \bar{S}$$

$$\overline{R \cup S} = \bar{R} \cap \bar{S}$$

8. **Law of excluded middle and law of contradiction are not satisfied.**

i.e. $R \cup \bar{R} \neq E_R$

and $R \cap \bar{R} \neq \phi_R$

Q. 11 Discuss fuzzy composition techniques with suitable example.

Ans. :

Composition operation can be used to combine two fuzzy relations in different product spaces.

There are two compositions that are used commonly.

1. **Max - Min Composition**

Assuming R_1 is a fuzzy relation defined on X × Y.

And R_2 is a fuzzy relation defined on Y × Z.

Then the max - min composition of two fuzzy relations R_1 and R_2 is denoted by $R_1 \circ R_2$ and defined as,

$$R_1 \circ R_2 = \{[(x, z), \max_{y \in Y} (\min (\mu_{R1}(x, y), \mu_{R2}(y, z))] \mid x \in X, y \in Y, z \in Z\}$$

OR

$$\mu_{R1 \circ R2}(x, z) = \max_{y \in Y} \{\min (\mu_{R1}(x, y), \mu_{R2}(y, z))\}$$

2. **Max - Product Composition**

The max - product composition is defined as,

$$R_1 \circ R_2 = \{[(x, z), \max_{y \in Y} (\mu_{R1}(x, y) \cdot \mu_{R2}(y, z))] \mid x \in X, y \in Y, z \in Z\}$$

OR

$$\mu_{R1 \circ R2}(x, z) = \max_{y \in Y} \{\mu_{R1}(x, y) \cdot \mu_{R2}(y, z)\}$$

The following are the properties of fuzzy composition. Assuming R, S and T are binary relations defined on X × Y, Y × Z and Z × W respectively.

1. Associativity → $R \circ (S \circ T) \Rightarrow (R \circ S) \circ T$
2. Monotonicity → $S \subseteq T \Rightarrow R \circ S \subseteq R \circ T$
3. Distributivity→ $R \circ (S \cup T) \Rightarrow (R \circ S) \cup (R \circ T)$
4. Inverse → $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$

Q. 12 Let R be the relation that specifies the relationship between the 'color of a fruit' and 'grade of maturity'. Relation S specifies the relationship between 'grade of maturity' and 'taste of a fruit', where color, grade and taste of a fruit are characterized by crisp sets x, y, z respectively as follows.

X = {green, yellow, red}

Y = {verdant, half mature, mature}

Z = {sour, tasteless, sweet}

Consider following relations R and S and find the relationship between 'color and taste' of a fruit using

1. **Max - min composition**
2. **Max - product composition**

R	Verdant	Half mature	Mature
Green	1	0.5	0
Yellow	0.3	1	0.4
Red	0	0.2	1

S	Sour	Tasteless	Sweet
Verdant	1	0.2	0
Half mature	0.7	1	0.3
Mature	0	0.7	1

Ans. :

1. Max - min composition

T	Sour	Tasteless	Sweet
Green	1	0.5	0.3
Yellow	0.7	1	0.4
Red	0.2	0.7	1

...Ans.

T (green, sour) = max (min (1, 1), min (0.5, 0.7), min (0, 0))
= max (1, 0.5, 0)
= 1

T (green, tasteless) = max (min (1, 0.2), min (0.5, 1), min (0, 0.7))
= max (0.2, 0.5, 0)
= 0.5

T (green, sweet) = max (min (1, 0), min (0.5, 0.3), min (0, 1))
= max (0, 0.3, 0)
= 0.3

T (yellow, sour) = max (min (0.3, 1), min (1, 0.7), min (0.4, 0.7))
= max (0.3, 0.7, 0.4)
= 0.7

T (yellow, tasteless) = max (min (0.3, 0.2), min (1, 1), min (0.4, 0.7))
= max (0.2, 1, 0.4) = 1

T (yellow, sweet) = max (min (0.3, 0), min (1, 0.3), min (0.4, 1))
= max (0, 0.3, 0.4)
= 0.4

T (red, sour) = max (min (0, 1), min (0.2, 0.7), min (1, 0))
= max (0, 0.2, 0) = 0.2

T (red, tasteless) = max (min (0, 0.2), min (0.2, 1), min (1, 0.7))
= max (0, 0.2, 0.7)
= 0.7

T (red, sweet) = max (min (0, 0), min (0.2, 0.3), min (1, 1))
= max (0, 0.2, 1)
= 1

2. Max - product composition

T	Sour	Tasteless	Sweet
Green	1	0.5	0.15
Yellow	0.7	1	0.4
Red	0.14	0.7	1

...Ans.

T (green, sour) = max (1 × 1, 0.5 × 0.7, 0 × 0)
= max (1, 0.35, 0)
= 1

T (green, tasteless) = max (1 × 0.2, 0.5 × 1, 0 × 0.7)
= max (0.2, 0.5, 0)
= 0.5

T (green, sweet) = max (1 × 0, 0.5 × 0.3, 0 × 1)
= max (0, 0.15, 0)
= 0.15

T (yellow, sour) = max (0.3 × 1, 1 × 0.7, 0.4 × 0.7)
= max (0.3, 0.7, 0.28)
= 0.7

T (yellow, tasteless) = max (0.3 × 0.2, 1 × 1, 0.4 × 0.7)
= max (0.06, 1, 0.28)
= 1

T (yellow, sweet) = max (0.3 × 0, 1 × 0.3, 0.4 × 1)
= max (0, 0.3, 0.4)
= 0.4

T (red, sour) = max (0 × 1, 0.2 × 0.7, 1 × 0)
= max (0, 0.14, 0)
= 0.14

T (red, tasteless) = max (0 × 0.2, 0.2 × 1, 1 × 0.7)
= max (0, 0.2, 0.7)
= 0.7

T (red, sweet) = max (0 × 0, 0.2 × 0.3, 1 × 1)
= max (0, 0.06, 1)
= 1

Q. 13 Explain standard fuzzy membership functions.

Ans. :

There are several different standard MFs available.

1. Increasing MFs (T Function)

An increasing MF is specified by two parameters (a, b) as follows:

$$T(x; a, b) = \begin{cases} 0 & ; \ x \le a \\ (x-a)/(b-a) & ; \ a \le x \le b \\ 1 & ; \ x \ge b \end{cases}$$

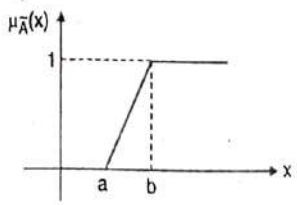


Fig. 2.8 : Increasing MF

2. Decreasing MF (L function)

A decreasing MF is specified by two parameters (a, b) as follows :

$$L(x; a, b) = \begin{cases} 1 & ; x \le a \\ (b-x)/(b-a) & ; a \le x \le b \\ 0 & ; x \ge b \end{cases}$$

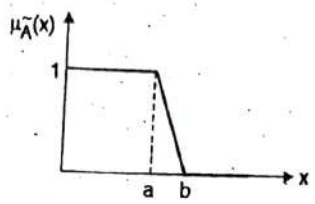


Fig. 2.9 : Decreasing MF

3. Triangular MF (∧ function)

A triangular MF is specified by three parameters (a, b, c) as follows :

$$\wedge(x; a, b, c) = \begin{cases} 0 & ; x \le a \\ (x-a)/(b-a) & ; a \le x \le b \\ (c-x)/(c-b) & ; b \le x \le c \\ 0 & ; x \ge c \end{cases}$$

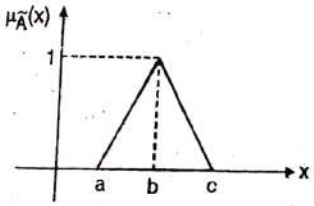


Fig. 2.10 : Triangular MF

4. Trapezoidal MFs (π function)

A trapezoidal MF is specified by four parameters (a, b, c, d) as follows :

trapezoid (x ; a, b, c, d) =

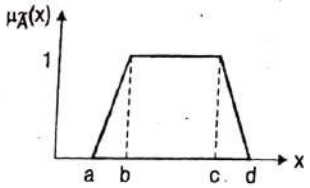


Fig. 2.7.5 : Trapezoid MF

An alternative expression using min and max can be given as,

$$\text{trapezoid}(x; a, b, c, d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right)$$

The parameters {a, b, c, d} (with $a < b < c < d$) determine the x coordinates of the four corners of the trapezoidal MF.

If b = c , then trapezoidal MF reduces to triangle MF.

Since two MFs triangular and trapezoidal are composed of straight line segment, they are not smooth at the corner points specified by the parameters. However due to the simple formulae and computational efficiency, they are used extensively.

Some **smooth** and **non-linear** MFs are as below:

5. Gaussian MFs

A Gaussian MF is specified by two parameters {c,σ}.

$$\text{Gaussian}(x; c, \sigma) = e^{-1/2\left(\frac{x-c}{\sigma}\right)^2}$$

c represents MFs center and

σ determines MFs width.

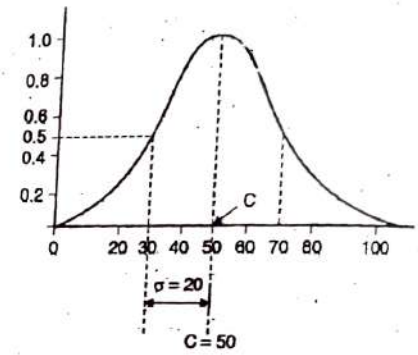


Fig. 2.11 : Gaussian (x; 50, 20) MF

6. Generalized bell MF / Cauchy MF

A generalized bell MF (or bell MF) is specified by three parameters (a, b, c).

$$\text{bell}(x; a, b, c) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}}$$

A desired generalized bell MF can be obtained by a proper selection of the parameters a, b, c.

c specifies the center of a bell MF

a specifies the width of a bell MF

and b determines the slope at the crossover points.

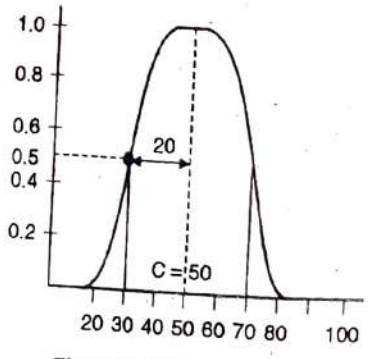
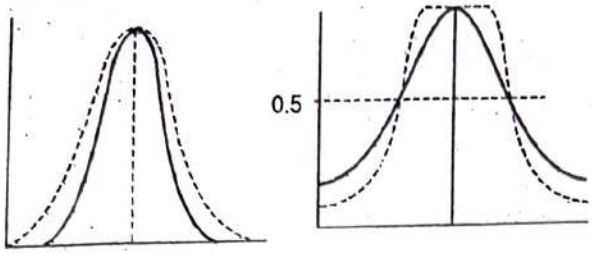
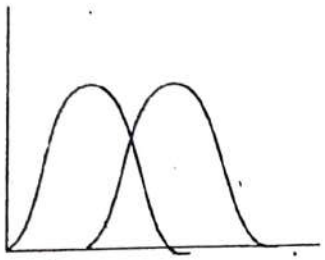


Fig. 2.12 : Bell (x ; 20, 4, 50)

Fig. 2.13 illustrates the effect of changing these parameters on the shape of the curve.



Changing 'a' (width) Changing 'b' (slope)



Changing 'c' (centre)

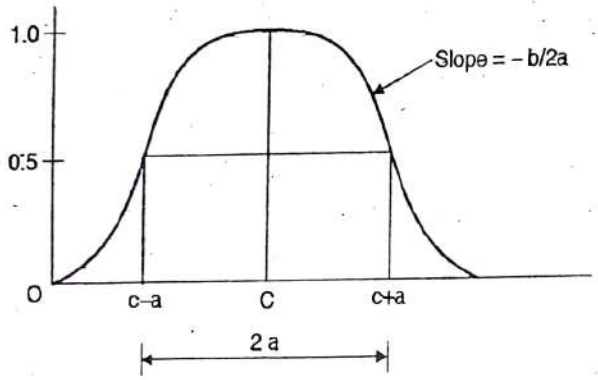


Fig. 2.13 : Effect of change of different parameters in Bell MF

The bell MF is direct generalization of Cauchy distribution used in probability theory; so it is also referred to as the Cauchy MF. The bell MF has more parameter than Gaussian MF, so it has more degree of freedom to adjust the steepness at the crossover point.

Although Gaussian and bell MFs achieves smoothness, they are unable to specify asymmetric MFs.

Asymmetric MFs

Asymmetric and close MFs can be achieved by using either the absolute difference or the product of two sigmoidal functions.

Sigmoidal MFs are as below,

7. Sigmoidal MFs

A sigmoidal MF is defined by,

$$\text{sig}(x ; a, c) = \frac{1}{1 + \exp[-a(x - c)]}$$

where, a controls the slope at the crossover point $x = c$.

Depending on the sign of the parameter a, a sigmoidal MF is open right or open left and thus is appropriate for representing concepts such as "very large" or "very negative".

They are widely used as the activation function in artificial neural networks.

Q. 14 Explain the features of membership function.

Ans. : Features of membership function

1. Support

A support of a fuzzy set $\tilde{A}$ is the set of all points x in X such that, $\mu_{\tilde{A}}(x) > 0$.

$$\text{Support}(\tilde{A}) = \{x \mid \mu_{\tilde{A}}(x) > 0\}$$

2. Core / Nucleus

The core of a fuzzy set $\tilde{A}$ is the set of all points x in X such that $\mu_{\tilde{A}}(x) = 1$.

$$\text{Core}\ \tilde{A} = \{x \mid \mu_{\tilde{A}}(x) = 1\}$$

3. Normality

A fuzzy set $\tilde{A}$ is normal if its core is nonempty. In other words there must be at least one point $x \in X$ such that $\mu_{\tilde{A}}(x) = 1$.

4. Crossover points

A cross over point of a fuzzy set $\tilde{A}$ is a point $x \in X$ at which $\mu_{\tilde{A}}(x) = 0.5$.

$$\text{Crossover}(\tilde{A}) = \{x \mid \mu_{\tilde{A}}(x) = 0.5\}$$

5. Fuzzy singleton

A fuzzy set whose support is a single point in X with $\mu_{\tilde{A}}(x) = 1$ is called a fuzzy singleton.

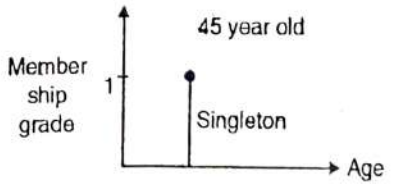


Fig. 2.14 : A fuzzy Singleton

Fig. 2.15 shows three parameters (core, support and crossover points) of a fuzzy set.

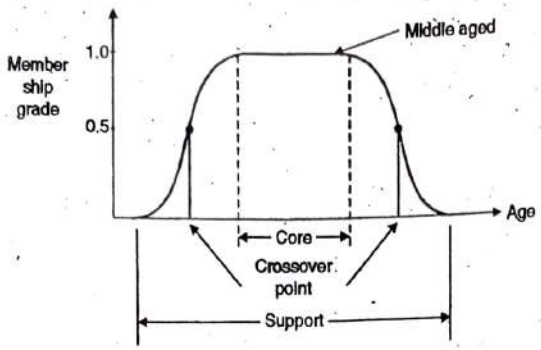


Fig. 2.15 : Core, Support and Crossover points of a fuzzy set

6. α-cut

The α-cut or α-level set of a fuzzy set $\tilde{A}$ is a crisp set defined by,

$$A_\alpha = \{x \mid \mu_{\tilde{A}}(x) \geq \alpha\}$$

7. Strong α-cut / strong α-level set

Strong α-cut is defined by

$$A'_\alpha = \{x \mid \mu_{\tilde{A}}(x) > \alpha\}$$

Using the above notations, support and core of a fuzzy set A can be expressed as,

$$\text{Support}(\tilde{A}) = A'_0 \qquad \text{Here } \alpha = 0$$

$$\text{Core}(\tilde{A}) = A_1 \qquad \text{Here } \alpha = 1$$

8. Convexity

A convex fuzzy set has membership function whose membership values are strictly monotonically increasing or strictly monotonically decreasing or strictly monotonically increasing than strictly monotonically decreasing with increasing values for elements in universe of discourse.

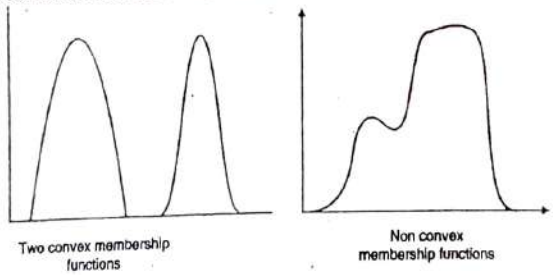


Fig. 2.16 : Convex and Non Convex MFs

Mathematically,

A fuzzy set $\tilde{A}$ is convex if and only if for any x_1 and $x_2 \in X$ and any $\lambda \in [0, 1]$.

$$\mu_{\tilde{A}}(\lambda x_1 + (1-\lambda) x_2) \geq \min\{\mu_{\tilde{A}}(x_1), \mu_{\tilde{A}}(x_2)\}$$

or

$\tilde{A}$ is convex if all its α-level sets are convex.

9. Fuzzy numbers

A fuzzy number $\tilde{A}$ is a fuzzy set in the real line (R) that satisfies the conditions for normality and convexity.

10. Bandwidth of normal and convex fuzzy set

For a normal and convex fuzzy set, the bandwidth or width is defined as the distance between two unique crossover points.

$$\text{Width}(\tilde{A}) = |x_2 - x_1|$$

Where $\mu_{\tilde{A}}(x_1) = \mu_{\tilde{A}}(x_2) = 0.5$

11. Symmetry

A fuzzy set $\tilde{A}$ is symmetric if its MF is symmetric around a certain point $x = c$,

$$\mu_{\tilde{A}}(c + x) = \mu_{\tilde{A}}(c - x) \quad \text{for all } x \in X$$

12. Open left, Open right and closed MFs

A fuzzy set $\tilde{A}$ is open left if,

$$\lim_{x \to -\infty} \mu_{\tilde{A}}(x) = 1 \text{ and}$$

$$\lim_{x \to +\infty} \mu_{\tilde{A}}(x) = 0$$

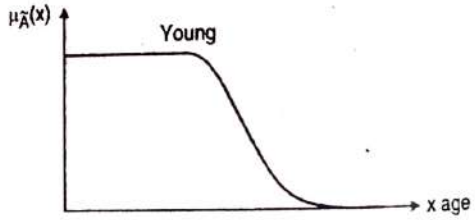


Fig. 2.17 : Open Left MF

A fuzzy set $\tilde{A}$ is open right if,

$$\lim_{x \to -\infty} \mu_{\tilde{A}}(x) = 0 \quad \text{and}$$

$$\lim_{x \to +\infty} \mu_{\tilde{A}}(x) = 1$$

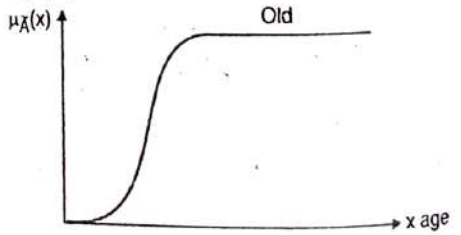


Fig. 2.18 : Open right MF

A fuzzy set $\tilde{A}$ is closed if,

$$\lim_{x \to +\infty} \mu_{\tilde{A}}(x) = 0 \quad \text{and}$$

$$\lim_{x \to -\infty} \mu_{\tilde{A}}(x) = 0$$

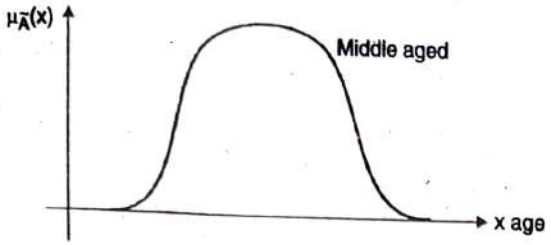


Fig. 2.19 : Closed MF

13. Cardinality

Cardinality of a fuzzy set $\tilde{A}$ is defined as,

$$|\tilde{A}| = \sum_{x \in X} \mu_{\tilde{A}}(x)$$

14. Relative Cardinality

Relative Cardinality of a fuzzy set $\tilde{A}$ is defined as,

$$\|\tilde{A}\| = \frac{|\tilde{A}|}{|X|}$$

15. Height of a fuzzy set

The height of a fuzzy set $\tilde{A}$ in X, is equal to the largest membership degree μ_m

$$\text{hgt}(\tilde{A}) = \sup_{x \in X} \mu_{\tilde{A}}(x)$$

if hgt $(\tilde{A}) = 1$ then, $\tilde{A}$ is normal

if hgt $(\tilde{A}) < 1$ then, $\tilde{A}$ is subnormal

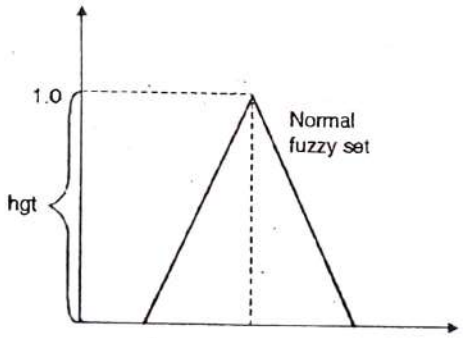


Fig. 2.20 : Height of a fuzzy set

Q. 15 Explain Inference in fuzzy logic.

Ans. : Inference in fuzzy logic

Fuzzy Inference System is the key unit of a fuzzy logic system. Fuzzy inference (reasoning) is the actual process of mapping from a given input to an output using fuzzy logic.

It uses the "IF...THEN" rules along with connectors "OR" or "AND" for drawing essential decision rules.

Fig. 2.21 (a) shows the block diagram of general fuzzy inference system.

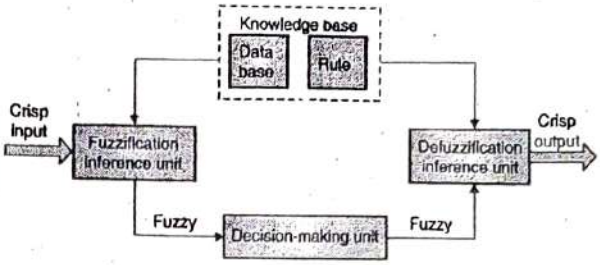


Fig 2.21(a) : Block diagram : Fuzzy inference system

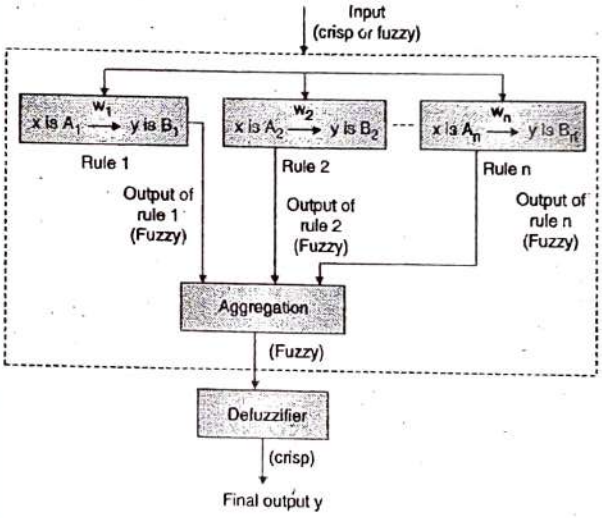


Fig 2.21 (b) : Fuzzy Inference using If-Then rules

As shown in Fig. 2.21 (a), FIS involves three important modules.

1. Fuzzification module (FM)
2. Decision making or Inferencing module
3. Defuzzification module (DM)

In addition to this, it uses two more components

- Data base and
- Rule base

} Knowledge base

Fuzzification module

This block performs a fuzzification which converts a crisp input in to a fuzzy set. Here we need to decide the proper fuzzification strategy.

Decision making/Inferencing

The basic function of the inference engine is to compute the overall value of the control output variable based on the individual contribution of each rule in the rule base. The output of the fuzzification module representing the crisp input is matched to each rule-antecedent.

The degree of match of each rule is established. Based on this degree of match, the value of the control output variable in the rule-consequent is modified. The result is, we get the "clipped" fuzzy set representing the control output variable. The set of all clipped control output values of the matched rules represent the overall fuzzy value of control output.

Defuzzification module

It performs defuzzification which converts the overall control output into a single crisp value.

The **rule base** and the **database** are jointly referred to as the **knowledge base**.

The **database** provides the necessary information for proper functioning of the fuzzification module, the rule base and the defuzzification module.

The information in the database includes :

1. Fuzzy MFs for the input and output control variables.
2. The physical domains of the actual problems and their normalized values along with the scaling factors.

Q. 16 Write short note on : fuzzy implications.

Ans. :

Fuzzy implications play an important role as logical connectives. They are used in fuzzy inference system where multiple criteria are to be combined for decision making. The most common logical connectives used in propositional logic (classical logic) are negation (NOT), conjunction (AND), disjunction (OR). In fuzzy logic theory, the t-norm and t-conorm operators can be considered as an abstraction of classical conjunction and disjunction. Fuzzy implication is a generalization of classical implications.

Classical implications

An implication in classical logic is a function: $I : \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}$, which satisfies the following boundary conditions.

(i) $I(0, 0) = 1$

(ii) $I(0, 1) = 1$

(iii) $I(1, 0) = 0$

(iv) $I(1, 1) = 1$

Fuzzy implications

Triangular norms and conorms are operations which generalize the logical conjunction and logical disjunction to fuzzy logic.

Triangular norms (T-Norms)

A triangular norm (t-norm) is a binary operation T on the interval [0, 1], satisfying the following conditions :

$T(x, y) = T(y, x)$ (commutativity)

$T(x, T(y, z)) = T(T(x, y), z)$ (associativity)

$y \leq z \Rightarrow T(x, y) \leq T(x, z)$ (monotonicity)

$T(x, 1) = x$ (neutral element 1)

Example of T-norms

$T_M(x, y) = \min(x, y)$ (minimum or Godel t-norm)

$T_P(x, y) = x \cdot y$ (product t-norm)

$T_L(x, y) = \max(x + y - 1, 0)$ (Lukasiewict t-norm)

Triangular co-norm (also called S-norm)

Its neutral element is 0 instead of 1, all other conditions remained same.

$S(x, y) = S(y, x)$ (commutativity)

$S(x, S(y, z)) = S(S(x, y), z)$ (associativity)

$y \leq z \Rightarrow S(x, y) \leq S(x, z)$ (monotonicity)

$S(x, 0) = x$ (neutral element 0)

Examples of t-conorm

$S_M(x, y) = \max(x, y)$ (maximum or Godel t-conorm)

$S_P(x, y) = x + y - x \cdot y$ (product t-conorm)

$S_L(x, y) = \min(x + y, 1)$ (Lukasiewict t-conorm , bounded sum)

Other implications

1. Kleene-Wienes implication

   $R(x, y) = \max(1 - x, y)$

2. The zadeh implication

   $R(x, y) = \max\{1 - x, \min(x, y)\}$

3. The Reichenbach implication

   $R(x, y) = 1 - x + xy$

4. The Willmott implication

$R(x, y) = \min\{\max\{1 - xy\}, \max\{1 - x, y\}, \max\{1 - y, y\}\}$

Q. 17 Write a short note on : Fuzzification.

Ans. :

Fuzzification is the process of converting a crisp set into a fuzzy set. Here the crisp value is transformed into linguistic variables. In real word problems, many a times the input values are not very precise and accurate rather they are uncertain, imprecise and unknown. The uncertainty may arise due to the vagueness and incompleteness of data. In such cases, variable may be represented as fuzzy and can be represented as fuzzy membership function.

Methods of membership value of assignment

1. Intuition

This method is based upon the common intelligence of human. The human develops membership functions based on their own understanding capability.

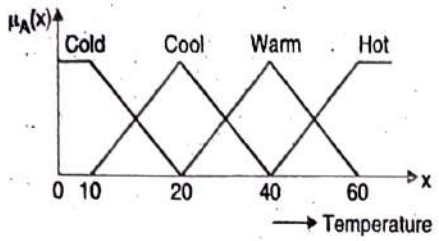


Fig. 2.22 : Membership functions for fuzzy variable "temperature"

As shown in Fig. 2.22 each triangular curve is a membership function corresponding to various fuzzy (linguistic) variables such as cold, coal, warm etc.

2. Inference

In inference method we use knowledge to perform deductive reasoning. To deduce or infer a conclusion, we use the facts and knowledge on that particular problem. Considering the example of Geometric shapes for the identification of a triangle.

And A, B, C are the interior angles of a triangle such that,

$A \geq B \geq C > 0°$ and $A + B + C = 180°$

For this purpose following five types of triangles are

1. R = Approximately Right-angle triangle
2. I = Approximately Isosceles triangle
3. E = Approximately Equilateral triangle
4. I · R = Isosceles Right-angle triangle
5. T = Other type of triangle

We can infer membership values for all those types of triangles through the method of inference because we possess the knowledge about the geometry of their shapes.

The membership values for five types of triangle can be defined as,

$$\mu_R(A, B, C) = 1 - \frac{1}{90°} |A - 90°|$$

$$\mu_I(A, B, C) = 1 - \frac{1}{60°} \min \{(A - B), (B - C)\}$$

$$\mu_E(A, B, C) = 1 - \frac{1}{80°} |A - C|$$

$$\mu_{I \cdot R}(A, B, C) = \mu_{I \cap R}(A, B, C)$$

$$= \min \{\mu_I(A, B, C), \mu_R(A, B, C)\}$$

$$\mu_T(A, B, C) = \overline{(R \cup I \cup E)} = \bar{R} \cap \bar{I} \cap \bar{E}$$

3. Rank ordering

In rank ordering method, preferences are assigned by a single individual, committee, a poll and other opinion methods can be used to assign membership values to fuzzy variables.

Here the preferences are determined by pair wise comparisons and they are used to determine ordering of the membership.

4. Angular fuzzy sets

Angular fuzzy sets differ from normal fuzzy sets only in their coordinate description. Angular fuzzy sets are defined on a universe of angles; hence they are of repeating shapes for every 2π cycles. Angular fuzzy sets are used in the quantitative description of the linguistic variables, which are known as "truth values".

Q. 18 Explain any four defuzzification methods with suitable Example.

Ans. :

Defuzzification is the process of converting a fuzzy set into a crisp value. The output of a fuzzy process may be union of two or more fuzzy membership functions. In that case there is need to find crisp value as a representative of the entire fuzzy MF.

Different methods of defuzzification are :

1. Max-membership principle / Height method
2. Centre of Area / Gravity (Centroid) Method
3. Centre of sum (COS)
4. Weighted average method
5. Mean-max membership (Middle of maxima)
6. Centre of largest area
7. First (or last) of maxima
8. Bisector method

1. Max-membership principle / Height method

This method is limited to peak output functions. It uses the individual clipped or scaled central outputs.

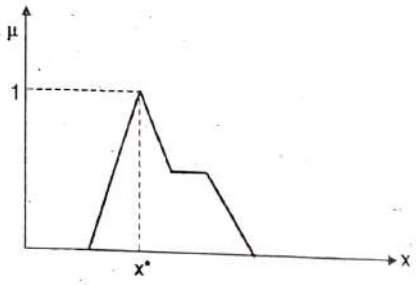


Fig. 2.23 : Max-membership

The algebraic expression is :

$$\mu_c(x^*) \geq \mu_c(x) \text{ for all } x \in X$$

2. Centre of Area / Gravity (Centroid) Method

This method is the most preferred and physically appealing of all the defuzzification methods. This method determines the centre of the area below the combined membership function. (i.e. it takes union of all output fuzzy sets). So, if there exist an overlapping area, will be considered only once. Thus overlapping areas are not reflected. This operation is computationally complex and therefore results in slow inference cycle.

Algebraic expression is

$x^* = \frac{\int \mu_c^-(x) \cdot x \, dx}{\int \mu_c^-(x) \, dx}$	$x^* = \frac{\sum_{i=1}^{n} \mu_c^-(x_i) \cdot x_i}{\sum_{i=1}^{n} \mu_c^-(x_i)}$

It is basically used for non-convex membership functions.

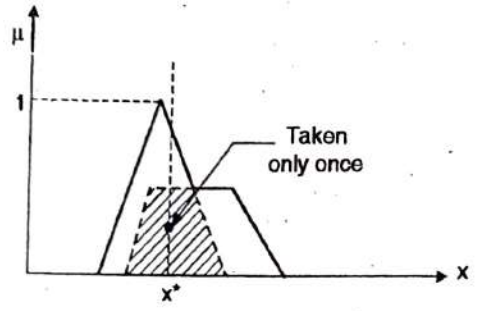


Fig. 2.24 : Centroid method

3. **Centre of sum (COS)**

This is faster than many defuzzification methods that are presently in use. This method involves the algebraic sum of individual output fuzzy sets, instead of their union. The idea is to consider the contribution of the area of each output membership curve. In contrast, the centre of area/gravity method considers the union of all output fuzzy sets. In COS method, we take overlapping areas. If such overlapping areas exist, they are reflected more than once.

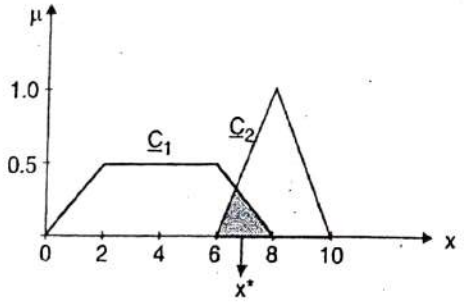


Fig. 2.25 : Centre of Sum method

For Continuous	For Discrete
$x^* = \frac{\int x \sum_{k=1}^{N} \mu_{C_k}^-(x) \, dx}{\int x \sum_{k=1}^{N} \mu_{C_k}^-(x) \, dx}$	$x^* = \frac{\sum_{i=1}^{N} x_i \sum_{k=1}^{n} \mu_{C_k}^-(x_i)}{\sum_{i=1}^{N} \sum_{k=1}^{n} \mu_{C_k}^-(x_i)}$

Advantage

It can be implemented easily and leads to a faster computation.

membership functions.

Algebraic expression is :

$$x^* = \frac{\sum_{i=1}^{n} \mu_c^-(x_i) \cdot xi}{\sum_{i=1}^{n} \mu_c^-(x_i)}$$

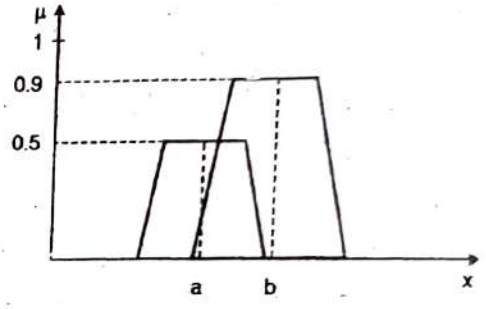


Fig. 2.26 : Weighted average method

The weighted average method is formed by weighting each membership function in the output by its respective maximum membership value. The two functions shown in Fig. 2.26 would result in the following general form of defuzzification.

$$x^* = \frac{(a \times 0.5) + (b \times 0.9)}{0.5 + 0.9}$$

Q. 19 Model the following as fuzzy set using suitable membership function. "Numbers close to 6".

Ans. :

Assuming universe of discourse is the set of all integer numbers.

$$X = \text{Integers}$$

Then fuzzy set "Numbers close to 6" can be defined as

$$\mu_A^-(x) = \frac{1}{1 + (x-6)^2} \quad \text{...Ans.}$$

Fig. 2.27 shows the plot of degree of membership of each element.

Table 2.1 : x and corresponding $\mu_A^-(x)$

x	$\mu_A^-(x)$
2	0.05
3	0.1
4	0.2
5	0.5
6	1
7	0.5
8	0.2
9	0.1
10	0.05

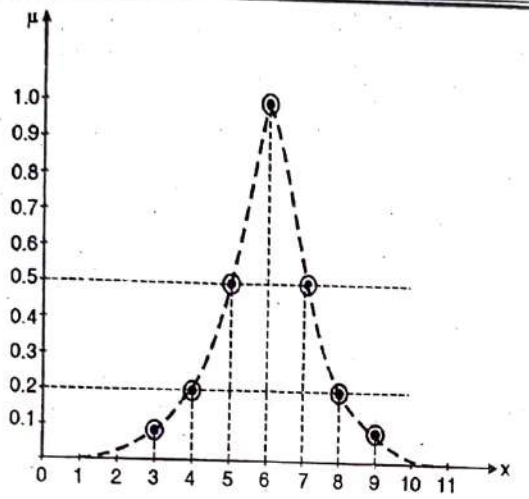


Fig. 2.27 : Plot of $x \rightarrow \mu_{\tilde{A}}(x)$

Q. 20 Determine all α-level sets and strong α-level sets for the following fuzzy set
A = {(1, 0.2), (2, 0.5), (3, 0.8), (4, 1), (5, 0.7), (6, 0.3)}

Ans. :

The following are α-level sets

$A_{0.2} = \{1, 2, 3, 4, 5, 6\}$

$A_{0.3} = \{2, 3, 4, 5, 6\}$

$A_{0.5} = \{2, 3, 4, 5\}$

$A_{0.7} = \{3, 4, 5\}$

$A_{0.8} = \{3, 4\}$

$A_1 = \{4\}$

Following are strong α-level sets.

$A_{0.2'} = \{2, 3, 4, 5, 6\}$

$A_{0.3'} = \{2, 3, 4, 5\}$

$A_{0.5'} = \{3, 4, 5\}$

$A_{0.7'} = \{3, 4\}$

$A_{0.8'} = \{4\}$

$A_{1'} = \phi$

Q. 21 Assume $\tilde{A}$ = "x considerably larger than 10" and $\tilde{B}$ = "x approximately 11"
characterized by $\tilde{A} = \{x, \mu_{\tilde{A}}(x) \mid x \in X\}$
Draw the plot for both the sets and show $\tilde{A} \cup \tilde{B}$ and $\tilde{A} \cap \tilde{B}$ in a plot.

Ans. :

Fuzzy set $\tilde{A}$ can be defined as,

$$\mu_{\tilde{A}}(x) = \begin{cases} 0 & , x \le 10 \\ \dfrac{1}{1 + \dfrac{1}{(x-10)^2}} & , x > 10 \end{cases}$$

Set $\tilde{B}$ can be defined as,

$$\mu_{\tilde{B}}(x) = \frac{1}{1 + (x-11)^2}$$

Then,

$$\mu_{\tilde{A} \cap \tilde{B}}(x) = \begin{cases} \min\left[(1 + (x-10)^{-2})^{-1}, (1 + (x-11)^2)^{-1}\right] & , x > 10 \\ 0 & , x \le 10 \end{cases}$$

That is, intersection operation on fuzzy set $\tilde{A}$ and $\tilde{B}$ represents a new fuzzy set "x considerably larger than 10 and approximately 11".

and

$$\mu_{\tilde{A} \cup \tilde{B}}(x) = \left\{ \max\left[(1 + (x-10)^{-2})^{-1}, (1 + (x-11)^2)^{-1}\right], \; x \in X \right.$$

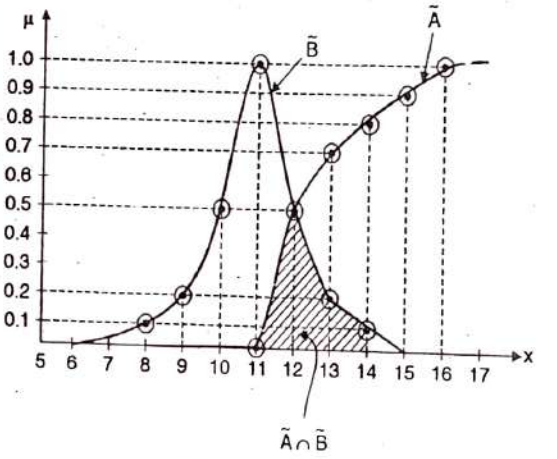


Fig. 2.28(a) : Plot of $\tilde{A} \cap \tilde{B}$

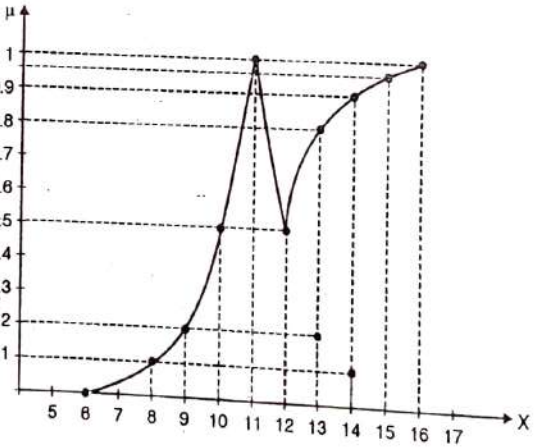


Fig. 2.28(b) : Plot of $\tilde{A} \cup \tilde{B}$

Q. 22 Let A = {a_1, a_2}, B = {b_1, b_2, b_3}, C = {c_1, c_2}. Let R be a relation from A to B defined by matrix.

	b_1	b_2	b_3
a_1	0.4	0.5	0
a_2	0.2	0.8	0.2

Let S be a relation from B to C defined by matrix.

	c_1	c_2
b_1	0.2	0.7
b_2	0.3	0.8
b_3	1	0

Find : (1) Max-min composition of R and S.
(2) Max-product composition of R and S.

Ans. :

(1) Max-min composition

$T = ROS =$

T	c_1	c_2
a_1	0.3	0.5
a_2	0.3	0.8

...Ans.

$T(a_1, c_1) = \max(\min(0.4, 0.2), \min(0.5, 0.3), \min(0, 1))$
$= \max(0.2, 0.3, 0) = 0.3$

$T(a_1, c_2) = \max(\min(0.4, 0.7), \min(0.5, 0.8), \min(0, 0))$
$= \max(0.4, 0.5, 0) = 0.5$

$T(a_2, c_1) = \max(\min(0.2, 0.2), \min(0.8, 0.3), \min(0.2, 1))$
$= \max(0.2, 0.3, 0.2) = 0.3$

$T(a_2, c_2) = \max(\min(0.2, 0.7), \min(0.8, 0.8), \min(0.2, 0))$
$= \max(0.2, 0.8, 0) = 0.8$

(2) Max-product composition

$T = ROS =$

T	c_1	c_2
a_1	0.15	0.4
a_2	0.24	0.64

...Ans.

$T(a_1, c_1) = \max(0.4 \times 0.2, 0.5 \times 0.3, 0 \times 1)$
$= \max(0.08, 0.15, 0) = 0.15$

$T(a_1, c_2) = \max(0.4 \times 0.7, 0.5 \times 0.8, 0 \times 0)$
$= \max(0.28, 0.40, 0) = 0.4$

$T(a_2, c_1) = \max(0.2 \times 0.2, 0.8 \times 0.3, 0.2 \times 1)$
$= \max(0.04, 0.24, 0.2) = 0.24$

$T(a_2, c_2) = \max(0.2 \times 0.7, 0.8 \times 0.8, 0.2 \times 0)$
$= \max(0.14, 0.64, 0) = 0.64$

Q. 23 High speed rail monitoring devices sometimes make use of sensitive sensors to measure the deflection of the earth when a rail car passes. These deflections are measured with respect to some distance from the rail car and, hence are actually very small angles measured in micro-radians. Let a universe of deflection be A = [1, 2, 3, 4] where A is the angle in micro-radians, and let a universe of distance be D = [1, 2, 5, 7] where D is distance in feet, suppose a relation between these two parameters has been determined as follows :

$R =$

	D_1	D_2	D_3	D_4
A_1	1	0.3	0.1	0
A_2	0.2	1	0.3	0.1
A_3	0	0.7	1	0.2
A_4	0	0.1	0.4	1

Now let a universe of rail car weights be W = [1, 2], where W is the weight in units of 100,000 pounds. Suppose the fuzzy relation of W to A is given by,

$S =$

	W_1	W_2
A_1	1	0.4
A_2	0.5	1
A_3	0.3	0.1
A_4	0	0

Using these two relations, find the relation R^T o S = T.

(a) Using max-min composition.

(b) Using max-product composition.

Ans. :

First find R^T

$R^T =$

	A_1	A_2	A_3	A_4
D_1	1	0.2	0	0
D_2	0.3	1	0.7	0.1
D_3	0.1	0.3	1	0.4
D_4	0	0.1	0.2	1

and $S =$

	W_1	W_2
A_1	1	0.4
A_2	0.5	1
A_3	0.3	0.1
A_4	0	0

(1) Using max-min composition

$T =$

	W_1	W_2
D_1	1	0.4
D_2	0.5	1
D_3	0.3	0.3
D_4	0.2	0.1

...Ans.

$T(D_1, W_1) = \max(1, 0.2, 0, 0) = 1$

$T(D_1, W_2) = \max(0.4, 0.2, 0, 0) = 0.4$

$T(D_2, W_1) = \max(0.3, 0.5, 0.3, 0) = 0.5$

$T(D_2, W_2) = \max(0.3, 1, 0.1, 0) = 1$

$T(D_3, W_1) = \max(0.1, 0.3, 0.3, 0) = 0.3$

$T(D_3, W_2) = \max(0.1, 0.3, 0.1, 0) = 0.3$

$T(D_4, W_1) = \max(0, 0.1, 0.2, 0) = 0.2$

$T(D_4, W_2) = \max(0, 0.1, 0.1, 0) = 0.1$

(2) Using max product composition

T =

	W_1	W_2
D_1	1	0.4
D_2	0.5	1
D_3	0.3	0.3
D_4	0.06	0.1

...Ans.

$T(D_1, W_1) = \max(1 \times 1, 0.2 \times 0.5, 0 \times 0.3, 0 \times 0)$

$= \max(1, 0.10, 0, 0) = 1$

$T(D_1, W_2) = \max(1 \times 0.4, 0.2 \times 1, 0 \times 0.1, 0 \times 0)$

$= \max(0.4, 0.2, 0, 0) = 0.4$

$T(D_2, W_1) = \max(0.3 \times 1, 1 \times 0.5, 0.7 \times 0.3, 0.1 \times 0)$

$= \max(0.3, 0.5, 0.21, 0) = 0.5$

$T(D_2, W_2) = \max(0.3 \times 0.4, 1 \times 1, 0.7 \times 0.1, 0.1 \times 0)$

$= \max(0.12, 1, 0.07, 0) = 1$

$T(D_3, W_1) = \max(0.1 \times 1, 0.3 \times 0.5, 1 \times 0.3, 0.4 \times 0)$

$= \max(0.1, 0.15, 0.3, 0) = 0.3$

$T(D_3, W_2) = \max(0.1 \times 0.4, 0.3 \times 1, 1 \times 0.1, 0.4 \times 0)$

$= \max(0.04, 0.3, 0.1, 0) = 0.3$

$T(D_4, W_1) = \max(0 \times 1, 0.1 \times 0.5, 0.2 \times 0.3, 1 \times 0)$

$= \max(0, 0.05, 0.06, 0) = 0.06$

$T(D_4, W_2) = \max(0 \times 0.4, 0.1 \times 1, 0.2 \times 0.1, 1 \times 0)$

$= \max(0, 0.1, 0.02, 0) = 0.1$

Q. 24 Model the following fuzzy set using trapezoidal membership function, "Middle age".

Ans. :

Assuming X is a reasonable age interval of human being.

$X = \{0, 1, 2, 3, \ldots, 100\}$

Then a fuzzy set "Middle age" can be represented using Trapezoidal MF as follows.

Trapezoid (x; 30, 40, 60, 70)

$$= \begin{cases} 0 & , \quad x \le 30 \\ (x-30)/10 & , \quad 30 \le x \le 40 \\ 1 & , \quad 40 \le x \le 60 \\ (70-x)/10 & , \quad 60 \le x \le 70 \\ 0 & , \quad x > 70 \end{cases}$$

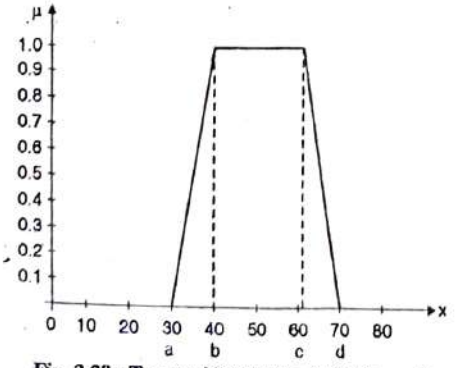


Fig. 2.29 : Trapezoidal MF for "Middle age"

Q. 25 For the given membership function as shown in Fig. 2.30, determine the defuzzified output value by any two methods.

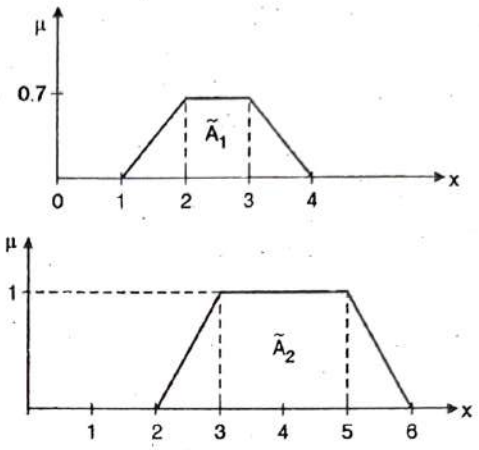


Fig. 2.30

Ans. :

1) Weighted Average method

In weighted average method , first the centre of each individual fuzzy set is calculated.

Centre of $\tilde{A}_1 = 2.5$ Centre of $\tilde{A}_2 = 4.0$

Next the membership value at the centre is calculated

Membership value of centre of $\tilde{A}_1 = 0.7$

Membership value of centre of $\tilde{A}_2 = 1$

$$x^* = \frac{(0.7 * 2.5) + (1 \times 4.0)}{0.7 + 1} = \frac{1.75 + 4}{1.7}$$

$= 3.38$ ...Ans.

2) Centre of sum method

The area of each individual curve is calculated.

As,

Area of Trapezoid =

$$\frac{[(\text{Sum of length of parallel lines}) \times (\text{distance between parallel lines})]}{2}$$

Therefore, Area of $\tilde{A}_1 = \frac{(1+3) * 0.7}{2} = 1.4$

Similarly, Area of $\tilde{A}_2 = \frac{(2+4) * 1}{2} = \frac{6}{2} = 3$

Next, center of each curve is calculated

Centre of $\tilde{A}_1 = 2.5$

Centre of $\tilde{A}_2 = 4$

$$X^* = \frac{(2.5 \times 1.4) + (4 \times 3)}{1.4 + 3} = \frac{15.5}{4.4} = 3.52$$ ...Ans.

Q. 26 Two fuzzy relations are given by

$$R = \begin{array}{c} \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{cc} y_1 & y_2 \end{array} \\ \begin{bmatrix} 0.6 & 0.3 \\ 0.2 & 0.9 \end{bmatrix} \end{array}$$

$$S = \begin{array}{c} \\ y_1 \\ y_2 \end{array} \begin{array}{c} \begin{array}{ccc} z_1 & z_2 & z_3 \end{array} \\ \begin{bmatrix} 1 & 0.5 & 0.3 \\ 0.8 & 0.4 & 0.7 \end{bmatrix} \end{array}$$

Obtain fuzzy relation T as a max-min composition and max-product composition between the fuzzy relations.

Ans. :

1. Using max-min composition

$$T(x_1, z_1) = \max(\min(0.6, 1), \min(0.3, 0.8)) = \max(0.6, 0.3) = 0.6$$

$$T(x_1, z_2) = \max(\min(0.6, 0.5), \min(0.3, 0.4)) = \max(0.5, 0.3) = 0.5$$

$$T(x_2, z_3) = \max(\min(0.6, 0.3), \min(0.3, 0.7)) = \max(0.3, 0.3) = 0.3$$

$$T(x_2, z_1) = \max(\min(0.2, 1), \min(0.9, 0.8)) = \max(0.2, 0.8) = 0.8$$

$$T(x_2, z_2) = \max(\min(0.2, 0.5), \min(0.9, 0.4)) = \max(0.2, 0.4) = 0.4$$

$$T(x_2, z_3) = \max(\min(0.2, 0.3), \min(0.9, 0.7)) = \max(0.2, 0.7) = 0.7$$

$$T = ROS = \begin{array}{c} \\ X_1 \\ X_2 \end{array} \begin{array}{c} \begin{array}{ccc} 0.6 & 0.5 & 0.3 \end{array} \\ \begin{bmatrix} 0.6 & 0.5 & 0.3 \\ 0.8 & 0.4 & 0.7 \end{bmatrix} \end{array} \text{...Ans.}$$

2. Using max-product

$$T(X_1, Z_1) = \max(0.6 \times 1, 0.3 \times 0.8) = \max(0.6, 0.24) = 0.6$$

$$T(X_1, Z_2) = \max(0.6 \times 0.5, 0.3 \times 0.4) = \max(0.30, 0.12) = 0.3$$

$$T(X_1, Z_3) = \max(0.6 \times 0.3, 0.3 \times 0.7) = \max(0.18, 0.21) = 0.21$$

$$T(X_2, Z_1) = \max(0.2 \times 1, 0.9 \times 0.8) = \max(0.2, 0.72) = 0.72$$

$$T(X_2, Z_2) = \max(0.2 \times 0.5, 0.9 \times 0.4) = \max(0.1, 0.36) = 0.36$$

$$T(X_2, Z_3) = \max(0.2 \times 0.3, 0.9 \times 0.7) = \max(0.06, 0.63) = 0.63$$

$$T = ROS = \begin{array}{c} \\ X_1 \\ X_2 \end{array} \begin{array}{c} \begin{array}{ccc} Z_1 & Z_2 & Z_3 \end{array} \\ \begin{bmatrix} 0.6 & 0.3 & 0.21 \\ 0.72 & 0.36 & 0.63 \end{bmatrix} \end{array} \text{...Ans.}$$

Chapter 3 : Fuzzy Systems

Q. 1 Explain the basic structure of fuzzy controller. And also explain the steps involved in designing the fuzzy controller.

Ans. :

Most commercial fuzzy products use fuzzy knowledge-based controllers (FKBC). The principal structure of a FKBC is shown in Fig. 3.1.

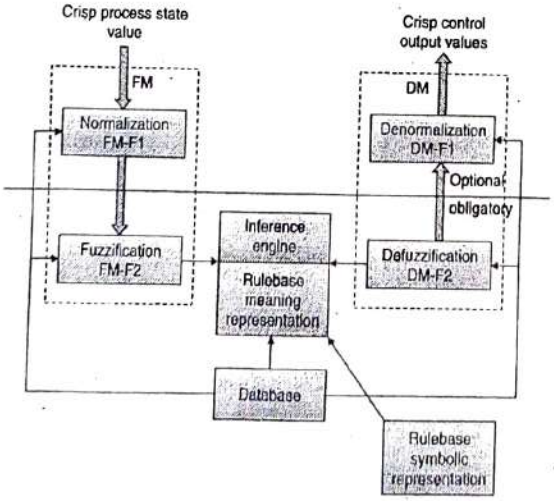


Fig. 3.1 : The structure of FKBC

As shown in Fig.3.1, FKBC involves three important modules.

1. Fuzzification module
2. Decision making or Inferencing module
3. Defuzzification module

In addition to this, it uses two more components

- Data base and
- Rule base

} Knowledge base

Fuzzification module

Fuzzification module performs the following two functions.

1. **Normalization**

This block performs a scale transformation which maps the physical values of input variables in to a normalized universe of discourse. This block is optional. If a non-normalized domain is used then this block is not required.

2. **Fuzzification**

This block performs a fuzzification which converts a crisp input in to a fuzzy set. Here we need to decide the proper fuzzification strategy.

Decision making/Inferencing

The basic function of the inference engine is to compute the overall value of the control output variable based on the individual contribution of each rule in the rule base. The output of the fuzzification module representing the crisp input is matched to each rule-antecedent.

The degree of match of each rule is established. Based on this degree of match, the value of the control output variable in the rule-consequent is modified. The result is, we get the "clipped" fuzzy set representing the control output variable. The set of all clipped control output values of the matched rules represent the overall fuzzy value of control output.

Defuzzification module

Defuzzification module performs two tasks :

1. Defuzzification

It performs defuzzification which converts the overall control output into a single crisp value.

2. Denormalization module

This block maps the crisp value of the control output into the physical domain. This block is optional. It is used only if normalization is performed during the fuzzification phase.

The knowledge base basically consists of a database and a rule base.

The **database** provides the necessary information for proper functioning of the fuzzification module, the rule base and the defuzzification module.

The information in the database includes :

1. Fuzzy MFs for the input and output control variables
2. The physical domains of the actual problems and their normalized values along with the scaling factors.

Steps in Designing FLC

Following are the steps involved in designing FLC

1. **Identification of variables :** Here, the input, output and state variables must be identified of the plant which is under consideration.
2. **Fuzzy subset configuration :** The universe of information is divided into number of fuzzy subsets and each subset is assigned a linguistic label. Always make sure that these fuzzy subsets include all the elements of universe.
3. **Obtaining membership function** : Now obtain the membership function for each fuzzy subset that we get in the above step.
4. **Fuzzy rule base configuration :** Now formulate the fuzzy rule base by assigning relationship between fuzzy input and output.
5. **Fuzzification :** The fuzzification process is initiated in this step.
6. **Combining fuzzy outputs :** By applying fuzzy approximate reasoning, locate the fuzzy output and merge them.
7. **Defuzzification** : Finally, initiate defuzzification process to form a crisp output.

Q. 2 State advantages of FLSs.

Ans. : Advantages of FLSs

1. It uses very simple Mathematical concepts for reasoning.
2. An FLS can be modified by just adding or deleting rules due to flexibility of fuzzy logic.
3. Fuzzy logic Systems can take imprecise, distorted, noisy input information.
4. FLSs are easy to construct and understand.

Q. 3 Explain disadvantages of FLSs.

Ans. : Disadvantages of FLSs

1. There is no systematic approach to fuzzy system designing.
2. They are understandable only when simple.
3. They are suitable for the problems which do not need high accuracy.

Q. 4 Explain different types of fuzzy propositions.

Ans. :

Propositions in fuzzy logic include the following :

1. Fuzzy predicate

Almost every predicate in natural language is fuzzy in nature. For example, tall, short, quick, hot etc.

2. Fuzzy-predicate Modifies

Fuzzy-predicate modifiers act as hedges for linguistic variables. For example, the words very, slightly, extremely are modifiers and the proposition can be "Water is very hot"

In the above example "hot" is a linguistic variable and "very" act as hedge.

3. Fuzzy Quantifier

A fuzzy quantifier is defined as a fuzzy number which provides an imprecise characterization of the cardinality of one or more fuzzy or non-fuzzy sets. Examples of fuzzy quantifiers are words like "most", "many", "few" etc. In other words, fuzzy quantifiers gives us an approximate idea of the number of elements of a subset fulfilling a certain conditions.

4. Fuzzy Qualifiers

A fuzzy qualifiers is also a proposition of fuzzy logic. There are four modes of qualifiers.

(a) Fuzzy truth qualification

It claims the degree of truth of a fuzzy proposition.

Expression : It is expressed as X is t. Here t is a fuzzy truth value.

Example : (Water is hot) is NOT VERY true. Here the qualified proposition is (water is hot) and the qualifying fuzzy truth value is "NOT VERY true"

(b) Fuzzy probability qualification

It claims the probability, either numerical or an interval, of fuzzy proposition

Expression : (Water is hot) is likely. Here the qualifying fuzzy probability is "likely".

(c) Fuzzy possibility qualification

It claims the possibility of fuzzy proposition.

Expression : It is expressed as "X is π", where π is a fuzzy possibility and can take values such as possible, quite possible, almost possible.

Example : (Water is hot) is almost impossible.

(d) Fuzzy usuality qualification

The propositions that are usually true or the events that have high probability of occurrence are related by the concept of usuality qualification.

Expression : "Usually (X is F)"

Here the subject X is a variable taking values in a universe of discourse U. Predicate F is a fuzzy subset of U and is interpreted as a usual value of X (denoted by U(X) = F).

Example : Usually Riya is very cheerful.

Q. 5 Explain how the formation and decomposition of compound rules occurs in Fuzzy logic.

Ans. : Formation of Rules

The general form of natural language expression is,

"IF antecedent THEN consequent."

The above expression is referred to as IF-THEN rules-based form.

There are three general forms for any linguistic variable.

a. Assignment statement

x = big

Tomato color = Red

Jack is not tall.

b. Conditional statement

IF temp is very cold THEN wear sweater. IF A is high THEN B is low ELSE C is low.

c. Unconditional statement

Goto sum

Stop

Multiply by 10

Turn the knob to the left

Decomposition of Compound Rules

A collection of many simple rules combined together is called a compound rule. It may be required to decompose such compound rules and reduce to a number of simple canonical rule forms.

Methods used for decomposition of rules

1. Multiple conjunctives antecedents

IF x is $\tilde{A}_1, \tilde{A}_2, \ldots, \tilde{A}_n$ THEN y is $\tilde{B}_m$. Assume a new fuzzy subset $\tilde{A}_m$ is defined as,

$$\tilde{A}_m = \tilde{A}_1 \cap \tilde{A}_2 \cap \ldots \cap \tilde{A}_n$$

and expressed as a membership function.

$$\mu_{\tilde{A}_m}(x) = \min \mu_{\tilde{A}_1}(x), \mu_{\tilde{A}_2}(x) \ldots \mu_{\tilde{A}_n}(x)$$

Which is based on the fuzzy intersection operation.

2. Multiple disjunctive antecedents

IF x is $\tilde{A}_1$ or x is $\tilde{A}_2$, ... OR x is $\tilde{A}_n$ THEN y is $\tilde{B}_m$ can be written as,

IF x is $\tilde{A}_n$ THEN y is $\tilde{B}_m$ where the fuzzy set $\tilde{A}_m$ is defined as,

$$\tilde{A}_m = \tilde{A}_1 \cup \tilde{A}_2 \cup \ldots \cup \tilde{A}_n \text{ and expressed as a membership function}$$

$$\mu_{\tilde{A}_m}(x) = \max \left[\mu_{\tilde{A}_1}(x), \mu_{\tilde{A}_2}(x), \ldots \mu_{\tilde{A}_n}(x)\right]$$

Which is based on the fuzzy union operation.

3. Conditional statement

The compound statement,

IF $\tilde{A}_1$ THEN ($\tilde{B}_1$ ELSE $\tilde{B}_2$) can be decomposed into two simple canonical rule forms, connected by OR.

IF $\tilde{A}_1$ THEN $\tilde{B}_1$

IF NOT $\tilde{A}_1$ THEN $\tilde{B}_2$

The compound statement,

IF $\tilde{A}_1$ (THEN $\tilde{B}_1$) UNLESS $\tilde{A}_2$ can be decomposed as,

IF $\tilde{A}_1$ THEN $\tilde{B}_1$

OR

IF $\tilde{A}_2$ THEN NOT $\tilde{B}_1$

The compound statement,

IF $\tilde{A}_1$ THEN $\tilde{B}_1$ ELSE IF $\tilde{A}_2$ THEN $\tilde{B}_2$

Can be decomposed as,

IF $\tilde{A}_1$ THEN $\tilde{B}_1$

OR

IF NOT $\tilde{A}_1$ AND IF $\tilde{A}_2$ THEN $\tilde{B}_2$

4. **Nested IF-THEN rules**

The rule,

"If $\tilde{A}_1$ THEN [IF $\tilde{A}_2$ THEN $\tilde{B}_1$]" can be decomposed as,

If $\tilde{A}_1$ AND $\tilde{A}_2$ THEN $\tilde{B}_1$

Q. 6 Write a note on different types of fuzzy approximate reasoning.

Ans. :

Fuzzy reasoning basically deals with reasoning that is approximate rather than fixed or exact. In fuzzy logic, both the antecedents and consequents are allowed to be fuzzy propositions. There exist four models of fuzzy approximate reasoning.

1. **Categorial Reasoning**

In this form of reasoning, the antecedent part of the rule does not contain any fuzzy quantifiers and fuzzy probabilities.

The antecedents are assumed to be in canonical form.

$\tilde{X}, \tilde{Y}, \tilde{Z}$ = Fuzzy variables taking in the universe U, V, W

$\tilde{X}, \tilde{B}, \tilde{C}$ = Fuzzy predicates

(a) The projection rule

The projection rule of inference is defined as,

$$\frac{\tilde{X}, \tilde{Y} \text{ is } \tilde{R}}{\tilde{X} \text{ is } [\tilde{R} \downarrow \tilde{X}]}$$

Where $[\tilde{R} \downarrow \tilde{X}]$ denotes the projection of fuzzy relation $\tilde{R}$ on $\tilde{X}$.

(b) The conjunction rule

The conjunction rule of inference is defined as,

$\tilde{X}$ is $\tilde{A}$, $\tilde{X}$ is $\tilde{B} \Rightarrow \tilde{X}$ is $\tilde{A} \cap \tilde{B}$

$(\tilde{X}, \tilde{Y})$ is $\tilde{A}$, $\tilde{X}$ is $\tilde{B} \Rightarrow (\tilde{X}, \tilde{Y})$ is $\tilde{A} \cap (\tilde{B} \times \tilde{V})$

$(\tilde{X}, \tilde{Y})$ is $\tilde{A}$, $(\tilde{Y}, \tilde{Z})$ is $\tilde{B} \Rightarrow (\tilde{X}, \tilde{Y}, \tilde{Z}) = (\tilde{A} \times \tilde{W}) \cap (\tilde{U} \times \tilde{B})$

(c) The disjunction rule

The disjunction rule of inference is defined as,

$\tilde{X}$ is $\tilde{A}$ OR $\tilde{X}$ is $\tilde{B} \Rightarrow \tilde{X}$ is $\tilde{A} \times \tilde{B}$

$\tilde{X}$ is $\tilde{A}$ OR $\tilde{Y}$ is $\tilde{B} \Rightarrow (\tilde{X}, \tilde{Y})$ is $\tilde{A} \times \tilde{B}$

(d) The negative rule of inference

NOT $(\tilde{X}$ is $\tilde{A}) \Rightarrow \tilde{X}$ is $\tilde{A}$

(e) The compositional rule of inference

$\tilde{X}$ is $\tilde{A}$, $(\tilde{X}, \tilde{Y})$ is $\tilde{R} \Rightarrow \tilde{Y}$ is $\tilde{A} \cdot \tilde{R}$

Where, $\tilde{A} \cdot \tilde{R}$ denotes the max-min composition of fuzzy set $\tilde{A}$ and fuzzy relation R and is given as,

$$\mu_{\tilde{A} \cdot \tilde{R}}(u) = \max_u \min \left[\mu_{\tilde{A}}(u), \mu_{\tilde{R}}(u, v)\right]$$

2. **Qualitative Reasoning**

In this mode of reasoning, the antecedents and consequents have fuzzy linguistic variables. The input-output relationship of a system is expressed as a collection of fuzzy IF-THEN rules. This reasoning is mainly used in control systems.

Assuming $\tilde{A}$ and $\tilde{B}$ are the fuzzy input variables and $\tilde{C}$ is the fuzzy output variable.

The relation between $\tilde{A}$ $\tilde{B}$ and $\tilde{C}$ may be expressed as,

If $\tilde{A}$ is x_1 AND $\tilde{B}$ is y_1, THEN $\tilde{C}$ is z_1

If $\tilde{A}$ is x_2 AND $\tilde{B}$ is y_2, THEN $\tilde{C}$ is z_2

⋮

If $\tilde{A}$ is x_n AND $\tilde{B}$ is y_n, THEN $\tilde{C}$ is z_n.

3. **Syllogistic Reasoning**

In this model, antecedents with fuzzy quantifiers are related to inference rules

A fuzzy syllogism can be expressed as,

$x = s_1$ A's are B's

$y = s_2$ C's are D's

$z = s_3$ E's are F's

Here, A, B, C, D, E, F are fuzzy predicates.

S_1 and S_2 are given fuzzy quantifiers.

S_3 is the fuzzy quantifier which has to be decided.

All fuzzy predicates provide a collection of fuzzy syllogism.

These syllogisms create a set of inference rules.

4. **Dispositional Reasoning**

In this type of reasoning, the antecedents are dispositions and may contain the fuzzy quantifier "usually". Usuality plays a major role here.

and working of FIS.

Ans. :

Construction and Working Principle of FIS

Fig. 3.2(a) shows the block diagram of general fuzzy inference system.

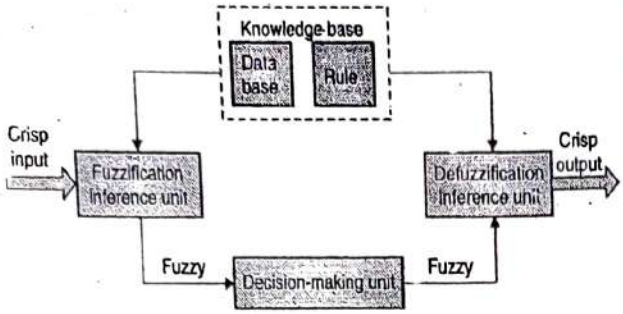


Fig. 3.2(a) : Block diagram : Fuzzy inference system

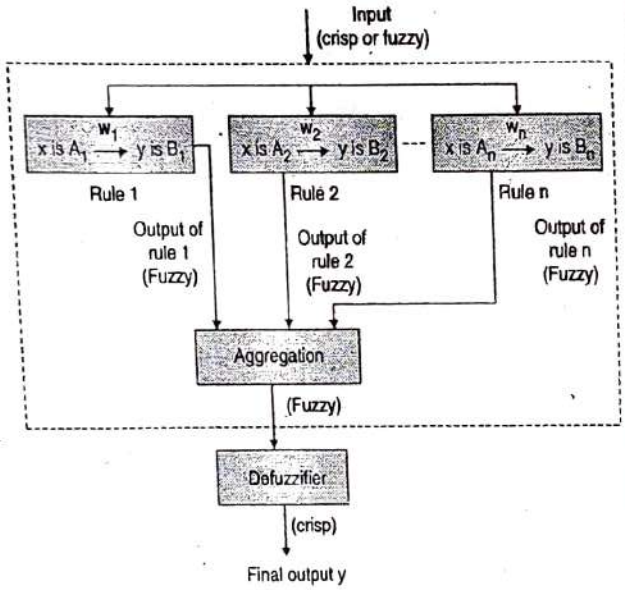


Fig 3.2(b) : Fuzzy Inference using If-Then rules

As shown in Fig. 3.2(a), FIS involves five important modules.

1. Fuzzification Inference Unit (FU)
2. Decision making / Inferencing Unit
3. Rule Base
4. Data Base
5. Defuzzification Inference Unit (DU)

Fuzzification Inference Unit

This block performs a fuzzification which converts a crisp input in to a fuzzy set. Here there is need to decide the proper fuzzification strategy.

Decision making/Inferencing Unit

The basic function of the inference unit is to compute the overall value of the control output variable based on the individual contribution of each rule in the rule base. The output of the each rule-antecedent.

The degree of match of each rule is established. Based on this degree of match, the value of the control output variable in the rule-consequent is modified. The result is, we get the "clipped" fuzzy set representing the control output variable. The set of all clipped control output values of the matched rules represent the overall fuzzy value of control output.

Defuzzification Unit

It performs defuzzification which converts the overall control output into a single crisp value. The rule base and the database are jointly referred to as the knowledge base.

A database

Data Base defines the membership functions of the fuzzy sets used in the fuzzy rules.

The information in the database includes:

1. Fuzzy Membership Functions for the input and output control variables
2. The physical domains of the actual problems and their normalized values along with the scaling factors.

A rule base

It contains a number of fuzzy IF-THEN rules;

Working

The input to the FIS may be a Fuzzy or crisp value.

1. Fuzzification Unit converts the crisp input into fuzzy input by using any of the fuzzification methods.
2. The next, rule base is formed. Database and rule base are collectively called knowledge base.
3. Finally, defuzzification process is carried out to produce crisp output.

Q. 8 How Sugeno model of FIS differ from Mamdani model ? Explain working of Mamdani FIS.

Ans. : Mamdani FIS

Mamdani FIS was proposed by Ebahim Mamdani in the year 1975 to control a steam engine and boiler combination. To compute the output of this FIS given the inputs, six steps has to be followed.

1. Determining a set of fuzzy rules.
2. Fuzzifying the inputs using the input membership functions.
3. Combining the fuzzified inputs according to the fuzzy rules to establish a rule strength (Fuzzy Operations).
4. Finding the consequence of the rule by combining the rule strength and the output membership function (implication).
5. Combining the consequences to get an output distribution (aggregation).

6. Defuzzifying the output distribution (this step is only if a crisp output (class) is needed).

Fuzzy Rule Composition in Mamdani Model

In Mamdani FIS, The fuzzy rules are formed using IF-THEN statements and AND/OR connectives.

The consequent of the rule can be obtained in two steps.

1. By computing the strength of each rule
2. By clipping the output membership function at the rule strength.

The outputs of all the fuzzy rules are then combined to obtain the aggregated fuzzy output. Finally, defuzzification is applied on to the aggregated fuzzy output to obtain a crisp output value.

Considering two inputs, two rule Mamdani fuzzy inference system. Assume two inputs are crisp value x and y.

Assuming the following two rules :

Rule 1 : if x is A_1 and y is B_1 then z is C_1

Rule 2 : if x is A_2 and y is B_2 then z is C_2

Fig. 3.3 shows Mamdani fuzzy inference system using **min – max** decomposition.

Fig. 3.3 illustrates a procedure of deriving overall output z when presented with two crisp inputs x and y. In the above Mamdani inference system, we have used **min** as T - norm and **max** as T - conorm operators.

The T-norm operator is used for inferencing antecedent part of the rule. And co-norm operator used to aggregate outputs resulting form of each rule.

Mamdani model also supports max - product composition to derive overall output z. Here the **algebraic product** is used as T-norm operator and **max** is used as T-conorm operator.

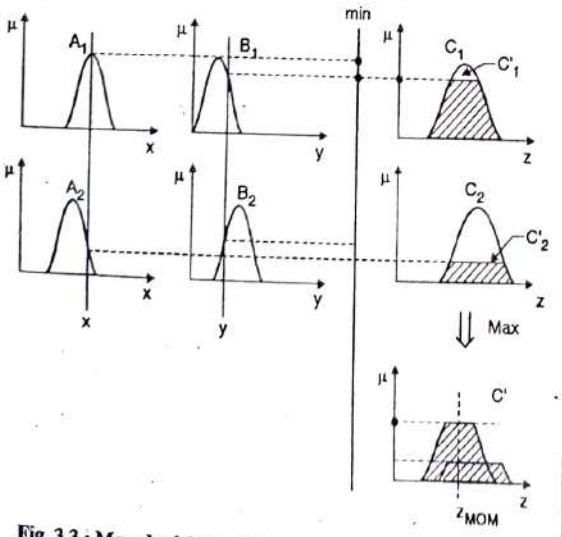


Fig. 3.3 : Mamdani fuzzy inference systems using max - min decomposition

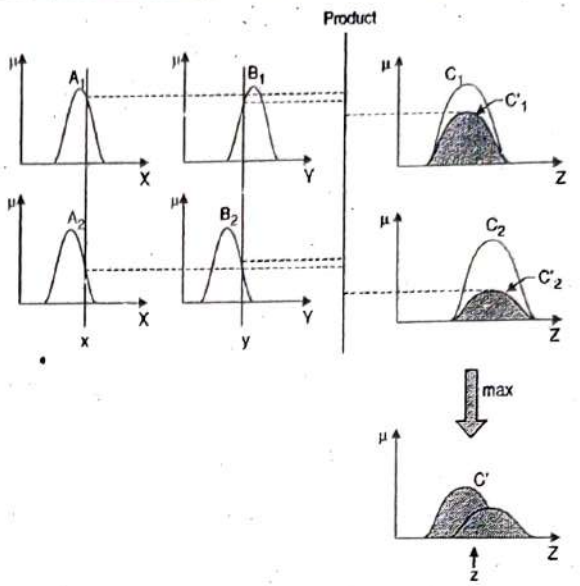


Fig. 3.3 : Mamdani fuzzy inference systems using max - product decomposition

Comparison between the Mamdani System and the Sugeno Model

1. **Output Membership Function :** The main difference between them is on the basis of output membership function. The Sugeno output membership functions are either linear or constant.
2. **Aggregation and Defuzzification Procedure :** The difference between them also lies in the consequence of fuzzy rules and due to the same their aggregation and defuzzification procedure also differs.
3. **Mathematical Rules :** More mathematical rules exist for the Sugeno rule than the Mamdani rule.
4. **Adjustable Parameters :** The Sugeno controller has more adjustable parameters than the Mamdani controller.

Q. 9 What is the form of Takagi-Sugeno-Kang (TSK) FIS ?

Ans. :

A typical fuzzy rule in TSK model has the form,

IF x is A and y is B then z = f(x, y)

Where,

x, y and z are linguistic variables.

A and B are fuzzy sets in the antecedent part of the rule.

Z = f(x, y) is a crisp function in the consequent part of the rule.

Usually f(x, y) is a polynomial in the input variables x and y.

Q. 10 Write a short note on : fuzzy expert system.

Ans. :

An expert fuzzy system is analogous to a human expert in a specific domain. An expert fuzzy system performs two major functions. It deals with uncertainty and incomplete/ vague information. It posses user-interaction function, which contains an explanation of system intention and desires. The fuzzy expert system incorporates fuzzy sets/ fuzzy logic for their reasoning process and knowledge representation scheme.

The basic block diagram of an expert system is shown in Fig. 3.4. It contains three main components.

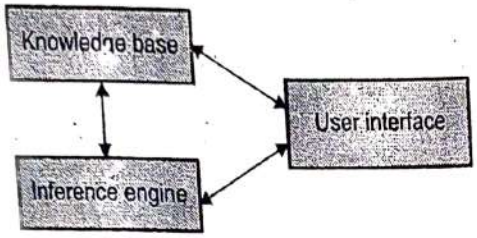


Fig 3.4 : Block diagram of an expert system

1. **Knowledge base :** contains the knowledge specific to the domain of application.
2. **Inference Engine :** It makes the use of knowledge present in the knowledge base to perform reasoning for user's queries.
3. **User Interface :** It's an interface where the user interacts with the system.

An example of an expert system is **MYCIN**, which introduces the concept of certainty factor for dealing with uncertainty. The rule strength in MYCIN is called certainty factor. This factor lies in the interval [0,1].When a rule is fired, its pre-state condition is evaluated. And a firing strength, is associated with the pre -state condition. When the rule is fired, the firing strength is compared with the previous mentioned threshold interval. If it is higher, the consequent of the rule is determined and a conclusion is made with a certainty.

The obtained conclusion and its certainty are the evidence provided by this fire rule for the hypothesis given by user. The hypothesis evidence from different rules is combined into belief measure and disbelief measures which are values lying in the [0,1] and [-1,0] respectively. If belief measure lies above a threshold, a hypothesis is believed, otherwise hypothesis is disbelieved.

Q. 11 Design a fuzzy controller to control the feed amount of purifier for the water purification plant. Raw water is purified by injecting chemicals. Assume input as water temperature and grade of water. Output as amount of purifier. Use three descriptors for input and output variables. Design rules to control action and defuzzification. Design should be supported by figures whenever necessary. Clearly indicate that when temperature is low, grade is low then chemical used is in large amount.

Ans. :

Step 1 : Identify input and output variables and decide descriptors for the same.

Here input variables are water temperature and grade of water. Water temperature is measured in °C. Grade of water is measured in percentage.

Descriptors for water temperature are

{C, M, H}

C - Cold

M - Medium

H - High

Descriptors for grade are

{L, M, H}

L - Low

M - Medium

H - High

Amount of purifier is measured in grams. Descriptors for amount of purifier are

{S, M, L}.

S - Small

M - Medium

L - Large

Step 2 : Fuzzification

Define membership functions for each of the input and output variables.

We use triangular MFs because of its simplicity.

(1) Membership functions for water temperature

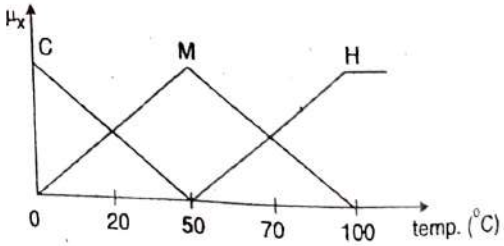


Fig. 3.5(a) : Membership functions for water temp.

$$\mu_C(x) = \frac{50 - x}{50}, \quad 0 \le x \le 50$$

$$\mu_M(x) = \begin{cases} \frac{x}{50}, & 0 \le x \le 50 \\ \frac{100 - x}{50}, & 50 < x \le 100 \end{cases}$$

$$\mu_H(x) = \frac{x - 50}{50}, \quad 50 \le x \le 100$$

(2) Membership functions for grade of water

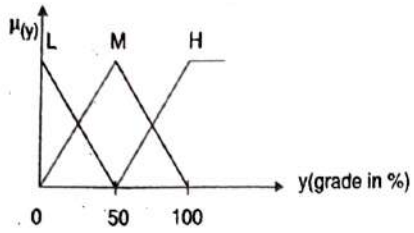


Fig. 3.5(b) : Membership functions for grade of water

$$\mu_L(y) = \frac{50-y}{50}, \quad 0 \le y \le 50$$

$$\mu_M(y) = \begin{cases} \frac{y}{50}, & 0 \le y \le 50 \\ \frac{100-y}{50}, & 50 < y \le 100 \end{cases}$$

$$\mu_H(y) = \frac{y-50}{50}, \quad 50 \le y \le 100$$

(3) Membership functions for amount of purifier

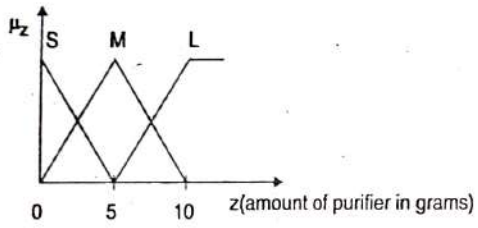


Fig. 3.5(c) : Membership functions for purifier

$$\mu_S(z) = \frac{5-z}{5}, \quad 0 \le z \le 5$$

$$\mu_M(z) = \begin{cases} \frac{z}{5}, & 0 \le z \le 5, \\ \frac{10-z}{5}, & 5 < z \le 10 \end{cases}$$

$$\mu_L(z) = \frac{z-5}{5}, \quad 5 < z \le 10$$

Step 3 : Form a Rule base

Temp \ grade	L	M	H
C	L	M	S
M	L	M	M
H	M	S	S

The above matrix represents in all nine rules. For example,

First rule can be, "If temperature is cold and grade is low then amount of purifier required is large."

Similarly all nine rules can be defined using if-then rules.

Step 4 : Rule Evaluation

Assume water temperature = 5° and grade = 30

Water temperature = 5° maps to the following two MFs of "temperature" variable.

$$\mu_c(x) = \frac{50-x}{50} \text{ and } \mu_M(x) = \frac{x}{50}$$

Similarly, grade = 30 maps to the following two MFs of "grade" variable.

$$\mu_L(y) = \frac{50-y}{50} \text{ and } \mu_M(y) = \frac{y}{50}$$

Evaluate $\mu_c(x)$ and $\mu_M(x)$ for x = 5°

To get,

$$\mu_c(5) = \frac{50-5}{50} = \frac{9}{10} = 0.9 \quad \ldots(1)$$

$$\mu_M(5) = \frac{5}{50} = \frac{1}{5} = 0.1 \quad \ldots(2)$$

Evaluating $\mu_L(y)$ and $\mu_M(y)$ for y = 30

$$\mu_L(30) = \frac{50-30}{50} = \frac{2}{5} = 0.4 \quad \ldots(3)$$

$$\mu_M(30) = \frac{30}{50} = \frac{3}{5} = 0.6 \quad \ldots(4)$$

The above four equation represents following for rules that we need to evaluate.

1. If temperature is **cold** and grade is **low**.
2. If temperature is **cold** and grade is **medium**.
3. If temperature is **medium** and grade is **low**.
4. If temperature is **medium** and grade is **medium**.

Since the antecedent part of each rule is connected by ***and*** operator we use ***min*** Operator to evaluate strength of each rule.

Strength of rule1 : $S_1 = \min(\mu_c(5), \mu_L(30))$

$= \min(0.9, 0.4) = 0.4$

Strength of rule2 : $S_2 = \min(\mu_c(5), \mu_M(30))$

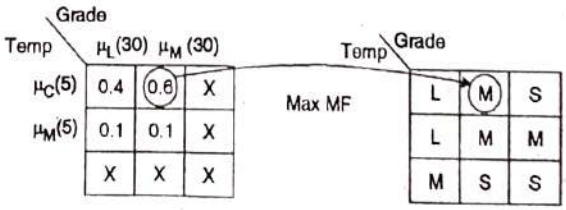
$= \min(0.9, 0.6) = 0.6$

Strength of rule3 : $S_3 = \min(\mu_M(5), \mu_L(30))$

$= \min(0.1, 0.4) = 0.1$

Strength of rule4 : $S_4 = \min(\mu_M(5), \mu_M(30))$

$= \min(0.1, 0.6) = 0.1$



(i) Rule strength table **(ii) Rule base table**

Fig. 3.5(d) : Rule strength and its mapping to corresponding output MF

Step 5 : Defuzzification

Since, we use "mean of max" defuzzification technique, we first find the rule with maximum strength.

$$= \max(S_1, S_2, S_3, S_4)$$

$$= \max(0.4, 0.6, 0.1, 0.1) = 0.6$$

This corresponds to rule 2.

Thus rule 2 : "Temperature is cold and grade is medium" has maximum strength 0.6.

The above rule corresponds to the output MF $\mu_M(z)$. This is show is shown in Fig. 3.5(d).

To find out final defuzzified value, we now take average (i.e. mean) of $\mu_M(z)$.

$$\mu_M(z) = \frac{10-z}{5} \quad \text{and} \quad \mu_M(z) = \frac{z}{5}$$

$$0.6 = \frac{10-z}{5} \qquad \therefore\ 0.6 = \frac{z}{5}$$

$$\therefore\ z = 13 \qquad \therefore\ z = 3$$

$$\therefore\ z^* = \frac{13+3}{2} = \mathbf{8\ gms} \qquad \text{...Ans.}$$

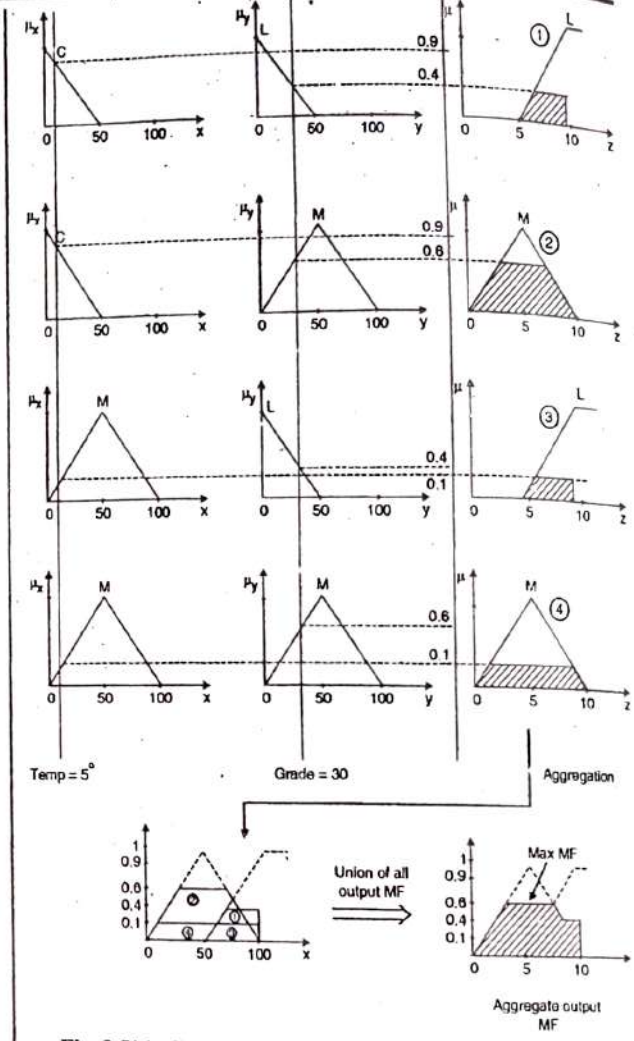


Fig. 3.5(e) : Process of rule evaluation and defuzzification

Chapter 4 : Evolutionary Computing

Q. 1 Explain a simple evolutionary system

Ans. :

Simulation of simple evolutionary system and its behavior over a time. Very first challenge here is, how to represent the individuals that make up an evolving population. One way to describe an individual as a fixed length vector of L features. These L features that are chosen should have potential to estimate an individual's fitness.

For example, individual might be represented as L = 5 following features.

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

The above vector specifies genetic makeup of an individual. i.e., its genotype specified as a chromosome with five genes (eye color, hair color, skin color, height, weight) whose values result in an individual with a particular trait of set. Alternatively, we could consider such vectors as description of the observable physical traits of individuals. i.e. their phenotype.

In addition to specifying a geno/phenospace, we need to define the "laws of motion" for an evolutionary system.

Consider the following pseudo code :

Simple EV

Generate an initial population of N individuals

Do forever

Select an individual from a current population to be a parent.

Use the selected parent to produce new offspring that is similar to but not the exact copy of the parent.

Select an individual of the population to die.

End Do

The above model is highly simplified model of the biological evolutionary systems.

Many things are yet not considered. For example, we have not considered distinction between genotype and phenotype. We have ignored the distinction between male and female and have only a sexual reproduction. A sexual production is ignored.

More Detailed EV is as follows :

EV

Randomly generate the initial population of N individuals (Using a uniform probability distribution over the entire geno/phenospace)

Compute the fitness of each individual in an initial population Do Forever

Choose a parent as follows :

Select a parent randomly using a uniform probability distribution over the current population.

Use the selected parent to produce a single offspring by :

Making an identical copy of the parent and then probabilistically mutating it to produce the offspring.

Compute the fitness of the offspring :

Select a member of the population to die by :

Randomly selecting a candidate for deletion from the current population using a uniform probability distribution and keeping either the candidate or the offspring depending on which one has higher fitness.

End Do

In the above algorithm, we compute the fitness of each individual. The fitness of the individual is computed using so called objective fitness. Also, offspring is produced by mutating a parent. We assume that each gene of an individual is equally likely to be muted. This means that on an average only one gene in every individual is muted. If there are L genes, each gene has an independent probability of 1/L of being selected to undergo a mutation.

Q. 2 Write a short note on : Evolutionary Systems as Problem Solvers.

Ans. :

The EV can be modified in many ways so that it would become more realistic model or natural evolutionary system.

Scientists design the systems with clear goals in mind, what functions to perform and what objectives to be met.

Computer scientists design and implement algorithms for sorting, searching, optimizing. Although EV system seems to be simple, it has potential to solve problems that require searching complex search space, solve hard optimization problem and are capable of adapting to changing environment.

Consider following simple change to EV.

EV

Generate an initial population of N individuals.

Do until a stopping criterion is met:

Select an individual from a current population to be a parent.

Use the selected parent to produce new offspring that is similar to but not the exact copy of the parent.

Select an individual of the population to die.

End Do

Return the individual with the highest global objective fitness.

The above algorithm adds a stopping criterion and returns an answer.

Q. 3 Classify Evolutionary Algorithms

Ans. :

Evolutionary algorithms are a heuristic-based approach to solve problems that cannot be easily solved in polynomial time, such as classically NP-Hard problems, and anything else that would take far too long to exhaustively process.

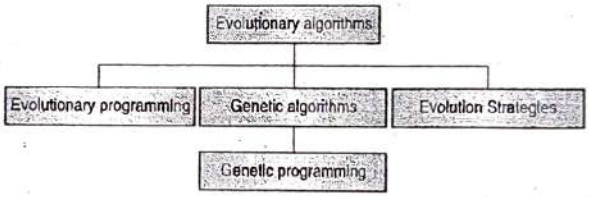


Fig. 4.1 : Classification of EA

Q. 4 Comment on the parent selection in EP.

Ans. :

EP considers phenotypic evolution.

EP iteratively applies two evolutionary operators :

1. Variation through application of mutation operators
2. Selection

In EP, M parents generates M offspring.

The chromosomes are represented as finite state machines (FSMs). The objective is to optimize these FSMs in order to provide a meaningful representation of behaviour based on the interpolation of the symbol. EP does not use **crossover** operators. Rather, the main process here is the mutation operation.

Here mutation is used to randomly create new offspring from the parent population. There exist five possible mutation operators :

1. Modify an output symbol.
2. Modify a state transition.
3. Add a state.
4. Delete a state.
5. Change the initial state.

The basic EP method involves 3 steps

(1) Choose an initial POPULATION of trial solutions at random.

(2) Each solution is replicated into a new POPULATION.

Each of these OFFSPRING solutions are mutated according to a distribution of MUTATION types.

(3) Each OFFSPRING solution is evaluated by computing it's FITNESS.

Q. 5 Explain the main components of evolutionary programming.

Ans. : The main components of an EP

Initialization

Initial population is generated by randomly selecting individual solutions

Evaluation

Fitness function measures the "behavioral error" of an individual with respect to the environment of that individual provides an absolute fitness measure of how well the problem is solved. Survival in EP is usually based on a relative fitness measure. A score is computed to quantify how well an individual compares with a randomly selected group of competing individuals. Individuals that survive to the next generation are selected based on this relative fitness. The search process in EP is therefore driven by a relative fitness measure, and not an absolute fitness measure.

Mutation

Mutation is the only means of variation in EP. (Crossover is not used at all)

Selection

Main purpose to select new population. A competitive process (Usually tournament selection) where parents and offspring compete to survive. Behaviors of individuals are influenced by strategy parameters.

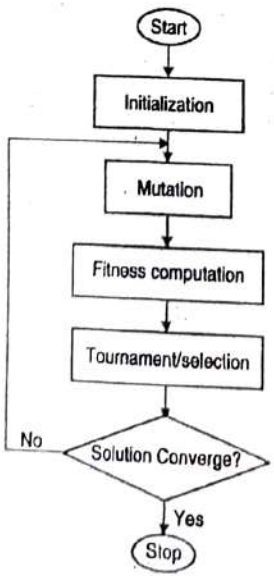


Fig. 4.2 : Flowchart : Evolutionary Programming

Mutation

Two important parameters, mutation operator and mutation step size need to be discussed.

Mutation operator

Mutation is the only way of introducing variations in EP.

In general, the mutation is defined as ,

$$X'_{ij}(t) = X_{ij}(t) + \Delta X_{ij}(t)$$

$X'_i(t)$ is the offspring and $\Delta X_i(t)$ is the mutational step size.

Mutational step size

Noise is sampled from some probability distribution.

Deviation of noise is determined by a strategy parameter, σ_{ij}

Generally, the step size is calculated as,

$$\Delta X_{ij}(t) = \emptyset(\sigma_{ij}(t)) \cdot \eta_{ij}(t)$$

Where, $\emptyset$ is a scaling function that scales the contribution of noise.

Based on the characteristics of scaling function $\emptyset$, EP can have following three categories.

1. **Non-adaptive EP :** The deviations in step sizes remain static.
2. **Dynamic EP :** Where the deviations in step sizes change overtime using some deterministic function, usually the function ϕ, a function of the fitness of individuals.
3. **Self-adaptive EP :** In which case deviations in step sizes change dynamically. The best values of σ_{ij} are learned in parallel with the decision variables, X_{ij}

Strategy Parameters

Deviation σ_{ij} are called strategy parameters.

Each individual has its own strategy parameter.

An individual is represented by a tuple.

$$X_i(t) = (X_i(t), \sigma_i(t))$$

Selection operator

The selection operator is used to create the new population. New population is selected from parents and their offspring. M represents the number of parents. Each individual in a parent set P(t) creates one child by mutation. So there will be exactly μ no. of children created. P(t) : μ parents, P'(t) : μ offspring

Pair-wise competition (tournament) is held in round-robin format :

Each solution x from $P(t) \cup P'(t)$ is evaluated against q other randomly chosen solutions - For each comparison, a "win" is assigned, if x is better than its opponent - The μ solutions with the greatest number of wins are retained to be parents of the next generation.

Q. 6 Comment on the parent selection in ES.

Ans. :

ES focuses on real valued parameter optimization. Individuals are represented as a vector of real-valued parameters.

Gaussian mutation operators is used as a reproduction operator.

M parent generates k >> M offspring.

Deterministic selection procedure is used to select the possible parents for the next generation.

Only best λ individuals are selected from a population P(s) that is of size μ.

And only these best λ individuals are used for the parent population Q(s).

This approach is called **$G(\mu, \lambda)$ strategy.**

If the λ best individuals are treated as elite that enters the next generation then the strategy is called $(\lambda + \mu)$ strategy.

Canonical versions of the ES

There are two variations of ES :

$(\mu / \rho, \lambda)$ – ES or alternatively (μ, λ) – ES (Comma notation) ...(1)

And $(\mu/\rho + \lambda)$ – ES or alternatively $(\mu + \lambda)$ – ES (Plus notation) ...(2)

Here μ is number of candidate solutions in the parent generation.

λ is the number of candidate solutions generated from the parent generation.

$\rho \leq \mu$ is the mixing number i.e., the number of parents involved in the creation of an offspring.

After creating λ offspring and calculating their fitness, the best μ of them are chosen deterministically, by either of the above mentioned selection method (1) or (2).

In the first method $(\mu/\rho, \lambda)$, parents are deterministically selected only form offspring.(Comma notation). In the second method$(\mu/\rho + \lambda)$, the parents are deterministically selected from both the parents and offspring (plus notation). Both the (μ, λ) and the $(\mu + \lambda)$ selection schemes are strictly deterministic and are based on rank rather than an absolute fitness value

Selection is based on the ranking of the individuals' fitness F(y) taking the μ best individuals. In general, an

ES individual a : = (y, s, F(y))

Comprises the *objective parameter vector* $y \in Y$ to be optimized.

A set of strategy parameters s.

The individual's observed fitness F(y) being equivalent to the objective function f(y), i.e., $F(y) \equiv f(y)$ in the simplest case.

The conceptual algorithm of the $(\mu/\rho, + \lambda)$ - ES is given below:

1. Initialize parent population $P_\mu = \{a_1,\ldots,a_\mu\}$.
2. Generate λ number of offspring a forming the offspring population $P_\lambda = \{a_1,\ldots, a_\lambda\}$ where each offspring a is generated by :
   (a) Select (randomly) ρ parents from P_μ
   (b) Recombine the ρ selected parents a to form a recombinant individual r.
   (c) Mutate the strategy parameter set s of the recombinant r.
   (d) Mutate the objective parameter set y of the recombinant r using the mutated strategy parameter.

Select new parent population (using deterministic truncation selection) from either

1. The offspring population P_λ or (comma notation)
2. The offspring P_λ and parent P_μ population (plus notation)

Goto 2. until *termination criterion* fulfilled.

Q. 7 Explain basic elements of a simple EA

Ans. :

A simple EA consists of the following basic elements:

1. Parent population size m,
2. Survival population size n
3. Parent selection methods
4. Survival selection methods

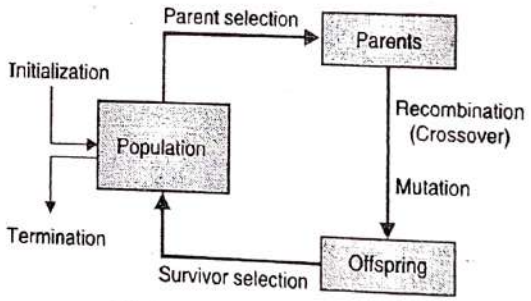


Fig 4.3 : A general schema of EA

1. Population Management

It involves managing parent population and children (offspring) population.

Two different population management models exist :

1. **Generational model :** Each individual survives for exactly one generation. The entire set of parents is replaced by the offspring.
2. **Steady-state model :** One offspring is generated per generation. One member of population replaced.

Generation Gap

The proportion of the population replaced.

Fitness based competition

Selection can occur in two places :

1. **Parent selection (selects mating pairs)** : Parent Selection is the process of selecting parents which mate and recombine to create off-springs for the next generation.
2. **Survivor selection (replaces population)** : 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 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.

Selection pressure

As selection pressure increases, fitter solutions are more likely to survive, or be chosen as parents

2. Parent Selection methods

a) Fitness Proportionate 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 and propagating their features to the next generation. Therefore, such a selection strategy applies a selection pressure to the more fit individuals in the population, evolving better individuals over time.

Considering a circular wheel. The wheel is divided into n pies, where n is the number of individuals in the population. Each individual gets a portion of the circle which is proportional to its fitness value.

Two implementations of fitness proportionate selection are possible :

Roulette Wheel Selection

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

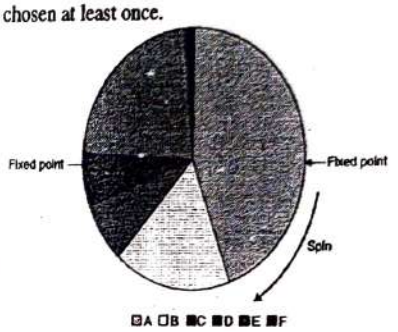
A fitter individual has a greater pie on the wheel and therefore a greater chance of landing in front of the fixed point when the wheel is rotated. Therefore, the probability of choosing an individual depends directly on its fitness.

Implementation wise, we use the following steps :

1. Calculate S = the sum of a finesses.
2. Generate a random number between 0 and S.
3. Starting from the top of the population, keep adding the finesses to the partial sum P, till P < S.
4. The individual for which P exceeds S is the chosen individual.

b) 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 points as shown in the following image. 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

Fig. 4.4

Fitness proportionate selection methods don't work for cases where the fitness can take a negative value.

c) Tournament Selection

In K-Way tournament selection, we select K individuals from the population at random and 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.

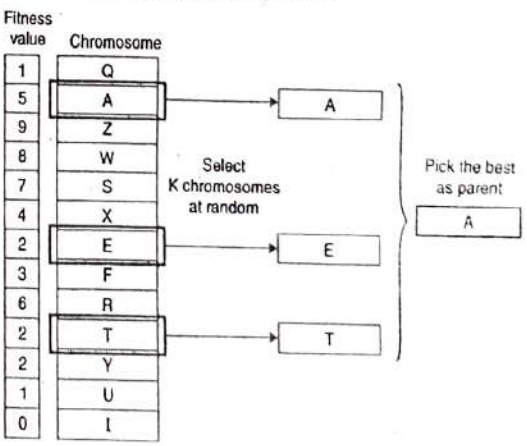
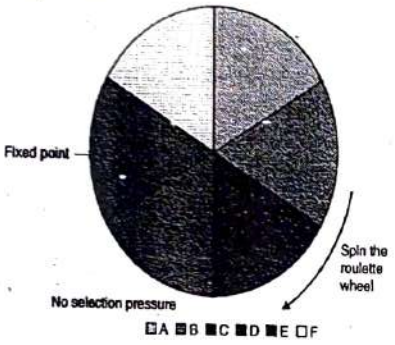


Fig. 4.5 : Example of tournament selection

d) Rank Selection

Rank Selection also works with negative fitness values and is mostly used when the individuals in the population have very close fitness values (this happens usually at the end of the run). This leads to each individual having an almost equal share of the pie (like in case of fitness proportionate selection) as shown in the following image and hence each individual no matter how fit relative to each other has an approximately 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.



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

Fig. 4.6 : Rank Selection

e) Uniform Selection

Parents are selected by uniform random distribution whenever an operator needs one/some. Uniform parent selection is unbiased - every individual has the same probability to be selected.

3. Survival Selection

Age Based Selection

In Age-Based Selection, we don't have a notion of fitness. It is based on the premise that each individual 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. For instance, in the following example, the age is the number of generations for which the individual has been in the population. The oldest members of the population i.e. P4 and P7 are kicked out of the population and the ages of the rest of the members are incremented by one.

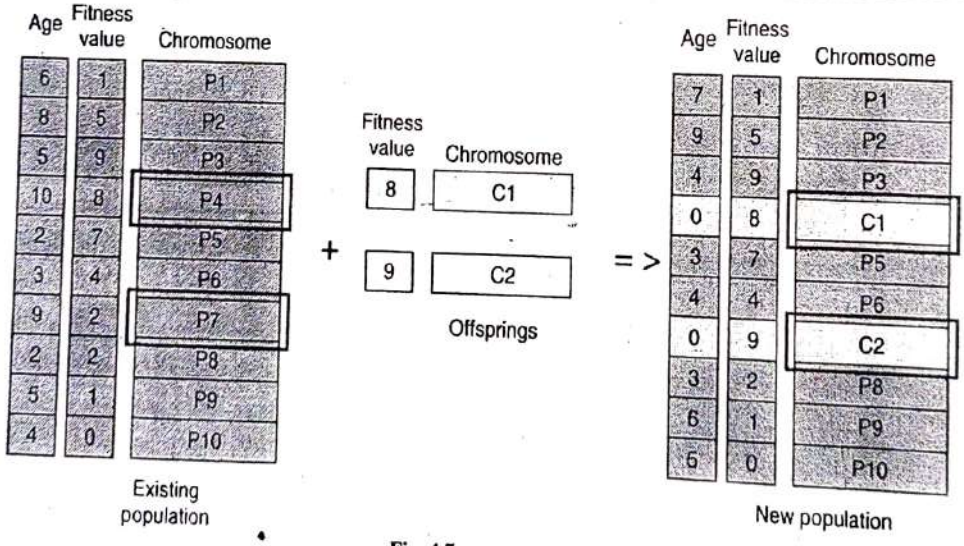


Fig. 4.7

Fitness Based Selection

In this fitness based selection, the children 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 image, the children replace the least fit individuals P1 and P10 of the population. It is to be noted that since P1 and P9 have the same fitness value, the decision to remove which individual from the population is arbitrary.

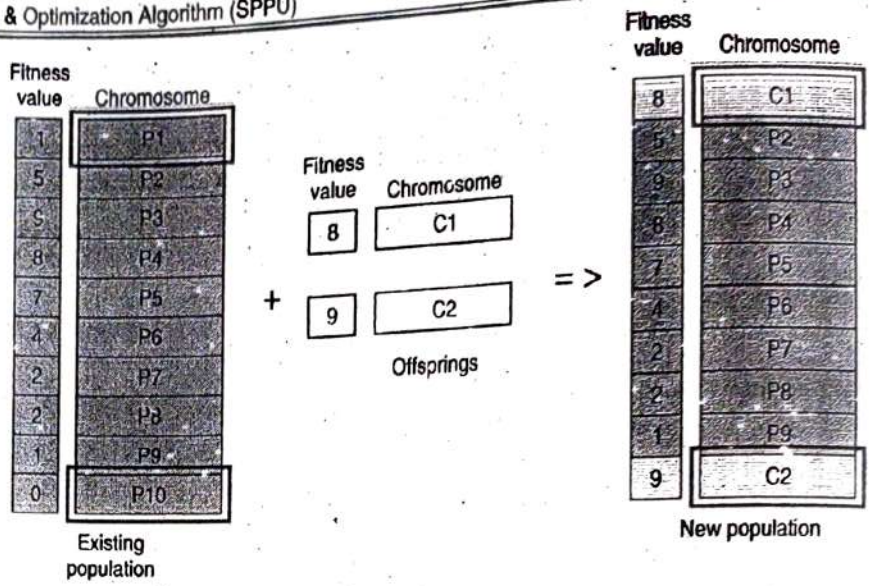


Fig. 4.8 : Fitness based selection

Elitism

In simple terms, it means the current fittest member of the population is always propagated to the next generation. Therefore, under no circumstance can the fittest member of the current population be replaced.

Q. 8 How genetic algorithms differ from evolutionary programming and evolutionary strategies?

Ans. :

Parameter	EP	ES	GA
Inventor	Fogel at California school in 1962.	Rechenberg and schwefel at German school in 1960s	Named by Holland at Michigan school in 1970s
Objective	To evolve intelligent behavioral model	Typically applied to numerical optimization.	To evolve genetic model
Representation	Individual solution is represented as a Finite State Machine (FMS).	The individuals are represented by Real-valued vector.	Individuals are represented as a binary string.
Parent selection	Parent selection in EP is stochastic using tournament	Parent selection in ES is Deterministic also called elitist selection. i.e. only the most fit individuals are allowed to reproduce.	Both deterministic and stochastic selection can be used.
Survival Selection	Probabilistic $(\mu + \mu)$ selection	(μ, λ) selection – selection based on the children only $(\mu + \lambda)$ – Selection based on both the set of parent and children	Children replace the parents
Mutation	Mutation in EP is Gaussian perturbation. Self-adaptation of mutation step size. The strategy parameters control the mutation of the object parameters.	Gaussian Mutation is used. Self-adaptation of mutation step size. The strategy parameters control the mutation of The object parameters.	GA mutation tend to be fixed size Mutation.
Recombination	EP does not use recombination to produce offspring. Rather it only uses mutation.	ES uses recombination such as cross over to produce offspring.	GA uses various recombination operators.

Q. 9 Is it advisable to apply genetic algorithm for all kinds of optimization problems? Justify.

Ans. :

No, it is not advisable to apply genetic algorithm for all kinds of optimization problems.

There is no single optimization algorithm that can solve all kinds of optimization problems. Which optimization technique is to be used, completely depends upon the kind of the problem being solved.

There are several questions need to be answered before choosing a right optimization technique such as,

1. Is problem linear?
2. Is it possible to calculate gradient?
3. Is your problem computationally intensive?
4. Is objective function continuous or discrete one?
5. Are there any constraint needs to be satisfied? Etc.

There is lots of optimization algorithms have been designed, and each of them suited for different tasks. For example, the simplex algorithm and its variants work great for linear programs, problems that can be expressed in terms of linear combinations of a set of continuous decisions. This approach is good for small problems.

Bigger linear programs often result from representational issues and a resulting explosion of decisions and constraints. Such problems can be still solved quite rapidly by simplex or interior point algorithms. Linear integer programming techniques like branch and bound or branch and cut often work better in these circumstances. Another option for problems with *discrete decisions that have nonlinear cost models* is dynamic programming, especially if there is a natural ordering to the decisions.

Genetic algorithms are slow, so they're not good for problem classes like the ones we described above, where special purpose algorithms exist specifically to solve them.

GAs are excellent for the problems that are hard to represent as a member of a known class of problems. For example, optimizing a statistic on the outputs of a simulation model over multiple random trials. There's no closed-form representation for this kind of problem. GAs are most appropriate for **complex non-linear** models where finding a location of the global optimum is a difficult task.

Multi-objective optimization is also a particular strength of genetic algorithms, because it takes advantage of the GA's population-based search. GAs, and especially MOEAs (Multi-objective evolutionary algorithms) are therefore great in an interactive decision-aiding context, where the user is free to tweak the problem and see what results emerge.

The list of topics to which genetic algorithms have been applied is extensive.

1. **Circuit optimization :** pick values of electronic components that meet timing constraints while minimizing leakage current.
2. **Production planning for manufacturing :** choose a product mix that balances risk and profit while minimizing overproduction, using probability models for yield and demand over time.
3. **Monitoring system design :** where to place sensors to minimize cost while maximizing coverage.

Chapter 5 : Genetic Algorithm

Q. 1 What are Genetic Algorithms?

Ans. :

Genetic Algorithms are ***adaptive heuristic search*** algorithms based on the evolutionary ideas of natural selection and genetics.

Evolution by natural selection is based on the principles of survival of the fittest by Charles Darwin. A GA can efficiently explore a large space of candidate designs and find optimum solutions.

Q. 2 Why Genetic Algorithms are used?

Ans. :

Genetic algorithms are basically based on the Darwinian's "survival of fittest" concept of natural selection and evolution to produce the solution for a given problem. Genetic algorithms are one of the best ways to solve a problem for which little information is known, especially, where the search space is very vast and non-linear. GAs are best suited for an environment in which there is a very large set of candidate solutions and in which the search space is uneven and has many hills and valleys.

Q. 3 Explain working principle of genetic algorithm.

Ans. :

To solve any problem, a genetic algorithm begins with a set of **solutions** (represented by **chromosomes**) called the **population.** Solutions from a particular population are taken and used to form a new population. The new population describes what we call the **next generation.**

Solutions are selected according to their **fitness** to form new solutions. Fitness is the ability of a solution to survive and pass on to the next generation. The more their fitness, the higher is their chance to reproduce.

The next generation (their offspring) are created by exchanging individual genes of parents selected and also by changing genes randomly in individual chromosomes. This is repeated until some condition (E.g. number of generations or improvement in the best solution) is satisfied.

Q. 4 Explain steps in genetic algorithm.

Ans. : Steps in Genetic Algorithm

Step 1 : Initialize a population with randomly generated individuals and evaluate the fitness value of each individual.

Step 2 :

(a) Select two individuals from the population with probability proportional to their respective fitness values.

(b) Apply crossover on the two selected individuals (in step(a)) with a probability equal to the crossover rate.

(c) Apply mutation with a probability equal to the mutation rate.

(d) Repeat (a) to (d) until enough members are generated to form the next generation.

Step 3 : Repeat steps 2 and 3 until the stopping criterion is met.

Flowchart of GA is depicted in Fig. 5.1.

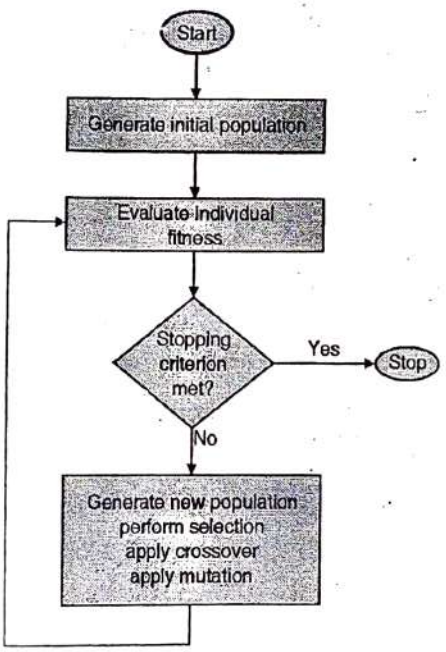


Fig. 5.1 : Steps in Genetic Algorithm

Each chromosome will be a function represented as a tree.

Q. 5 Explain different selection methods used in GA.

Ans. :

Selection is the process of selecting two or more parent chromosomes from a given generation of population for mating. After deciding on an encoding scheme, the next step is to decide how to perform selection i.e. choosing chromosomes in the current generation that will create offspring for the next generation.

In general, selection strategies must favour fitter candidates over weaker candidates.

The most commonly used strategies are :

1. Roulette Wheel Selection

In this selection scheme, parents are selected according to their fitness values. The higher the fitness value of a chromosome, the more is the chance that it will be selected. Conceptually, each chromosome of the population is allocated a section of an imaginary roulette wheel.

Unlike a real roulette wheel, the sections are of different sizes, proportional to the chromosome's fitness, such that the fittest candidate has the biggest slice of the wheel and the weakest chromosome has the smallest. The wheel is then spun and the chromosome associated with the winning section is selected. This process is repeated as many times as necessary to select the entire set of parents for the next generation.

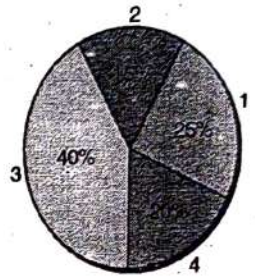


Fig. 5.2 : Roulette Wheel Selection

Fig. 5.2 shows the Roulette - Wheel for 4 chromosomes according to their fitness. Since the 3rd chromosome has a higher fitness than any other chromosome, it is expected that the roulette-wheel selection will choose it more than any other chromosome.

If $\left(\bar{F} = \sum_{j=1}^{n} F_j / n \right)$ is the average fitness of the population, then the Roulette-wheel selection is expected to make $F_j / \bar{F}$ copies of the j^{th} string in the population.

Where F_j is the fitness value of j^{th} chromosome and 'n' is the number of chromosomes in a chosen population.

2. Tournament Selection

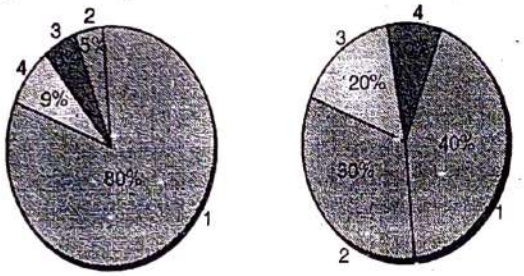
In tournament selection, two chromosomes from the population are randomly picked and a tournament is staged to determine which one gets selected. The winner of the "tournament" may be selected directly as the chromosome having the higher fitness among the two or it may be selected probabilistically.

The probabilistic tournament selection involves generating a random value between 0 and 1 and comparing it to a pre-determined selection probability. If the random value is less than or equal to the selection probability, the fitter candidate is selected, otherwise the weaker candidate is selected. The tournament can be extended to involve more than two individuals if desired.

3. Rank Selection

Rank selection attempts to remove problems of roulette wheel selection by basing selection probabilities on relative rather than absolute fitness. The problem with roulette wheel selection is that when fitness values differ very much, then it is biased towards the chromosome having the highest fitness and other chromosomes have very few chances to be selected. Rank selection first ranks the population and then every chromosome receives fitness from this ranking.

The worst chromosome will have fitness 1, 2nd worst will have fitness 2 and the best will have fitness 'n' (Number of chromosomes in the population). After the fitness has been allocated, the chromosomes are then selected using the same mechanism of roulette wheel selection.



(a) Situation before ranking (b) Situation after ranking

Fig. 5.3 : Rank Selection

As shown in Fig. 5.3, after ranking step of rank selection, all chromosomes have a good chance to be selected because the probability of selection is as per relative rather than absolute fitness values.

4. Steady - State Selection

Here, the goal is to allow a major part of the chromosomes to survive and pass to the next generation. As part of this strategy, in every generation, a few good (high fitness) chromosomes are selected for creating new offspring. Then, some bad (low fitness) chromosomes are removed and the new offspring are placed in their place. The rest of the population survives to the next generation.

5. Elitism

In elitism, the best (or a few best) chromosome(s) are copied directly to the next generation. The remaining chromosomes are then selected using one of the above strategies. Elitism ensures that good chromosomes are not lost and can rapidly increase the performance of GA.

Q. 6 Explain types of crossover operators.

Ans. :

The crossover is applied to the selected pair of parents with a hope that it would create a better string. The aim of the crossover operator is to search the parameter space. Different types of crossover operators can be used.

1. Single - Point Crossover

Here, a crossover point on the strings to be mated is selected at random and bits next to the crossover point are exchanged. This is shown in Fig. 5.4.

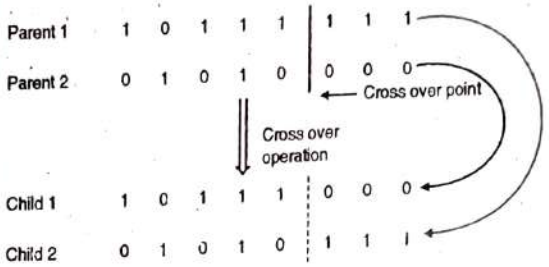


Fig. 5.4 : Single-point crossover

2. Two-Point Cross-over

In two point crossover, two crossover points are chosen at random and the parts of the chromosome strings between these two points is swapped. This is illustrated in Fig. 5.5.

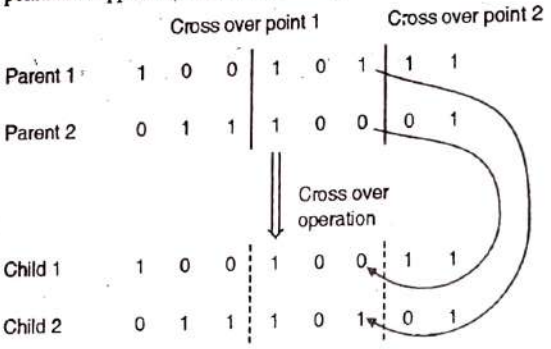


Fig. 5.5 : Two-point crossover

3. Multipoint Crossover

Multipoint crossover uses more than two crossover points. There could be two cases again. We can have even number of crossover points or odd number of crossover points. In even numbered cross-points, the string is treated as a ring with no beginning or end. The cross-points are selected around the circle uniformly at random. The information between alternate pairs of cross-points is interchanged as shown in Fig. 5.6.

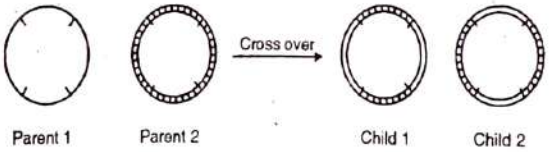


Fig. 5.6 : Multipoint cross over with even number of cross-points

In case of odd number of cross-points, a different cross-point is always assumed at the beginning of the string, the information (genes) between alternate pairs is then exchanged as shown in Fig. 5.7.

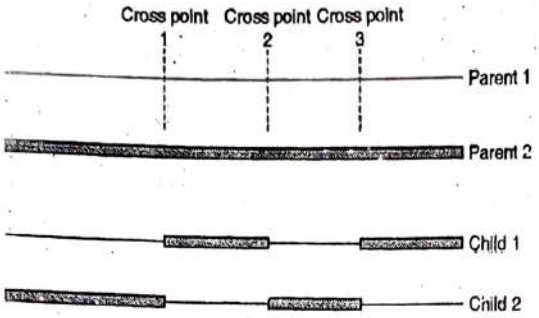


Fig. 5.7 : Multipoint crossover with odd number of cross-points

4. Uniform Crossover

In a uniform crossover operation, each bit from either parent is selected with a probability of 0.5 and then interchanged as shown in Fig. 5.8.

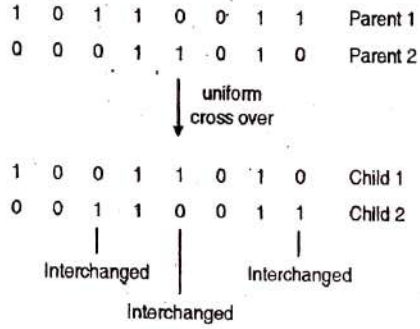


Fig. 5.8 : Uniform crossover

Another variation of uniform crossover is to define a crossover mask. Whenever there is a 1 in the mask, the gene is copied from the first parent and when there is 0 in the mask, the gene from the second parent is copied to produce the first offspring.

To produce the second offspring, the parents are exchanged. That is, parent 1 now becomes parent 2 and vice versa. This is illustrated in Fig. 5.9. A new mask is generated for each pair of parents.

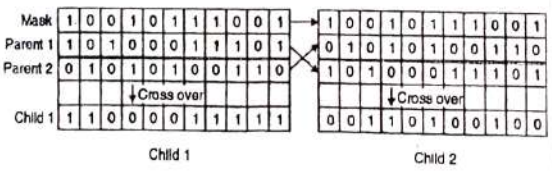


Fig. 5.9 : Uniform crossover using mask

5. Matrix Crossover (2D crossover)

In case of matrix crossover, each individual is represented as a 2D array of vector. The process of 2D crossover is depicted in Fig. 5.10.

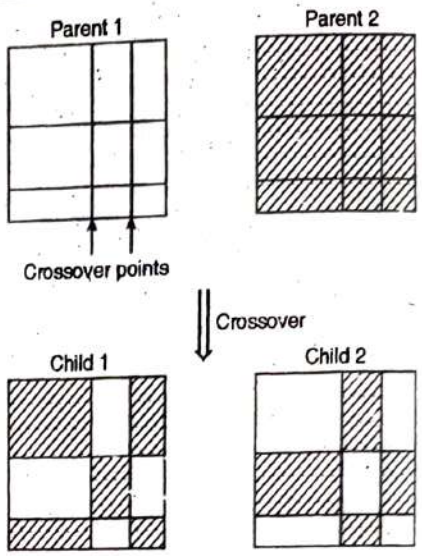


Fig. 5.10 : Matrix crossover

Here two random crossover points along row and column are chosen, then string is divided into non overlapping rectangular regions. As shown in Fig. 5.10, two crossover points both row wise and column wise divide the rectangle into nine regions.

Select any region in each layer, either vertically or horizontally and then exchange the information in that region between the mated populations.

Q. 7 What are the various types of mutation techniques used in GA?

Ans. :

A mutation is a permanent change in the sequence of DNA. After crossover, usually we perform mutation. Mutations can be advantageous and lead to an evolutionary advantage of a certain genotype.

There are different types of mutations :

1. Point Mutation

In case of binary encoding, point mutation is a process of flipping a bit in a chromosome, that is, changing a bit from 0 to 1 and vice versa. Whether the particular bit is to be flipped or not, that is decided by flipping a coin with a probability of P_m.

In coin flipping method, for a particular bit, a number between 0 and 1 is chosen at random. If the random number is smaller than P_m then that bit is flipped, otherwise it is kept unchanged. The bits of the string are independently mutated, that is the mutation of one bit does not affect probability of mutation of other bits.

Mutation is basically used to restore lost genetic material. The mutation operator introduces new genetic structures in the population by randomly modifying some of its building blocks. It helps the search algorithm to escape from the local minima's trap.

2. Replace

In this type of mutation, a single mutation point is generated randomly. Next, a gene is generated randomly. The gene at the mutation point is then replaced with the randomly generated gene to form the new chromosome.

Considering the following chromosome.

1 2 3 4 5 6 7 8 9

Assuming that the random mutation point generated was 4. Further assume that the random gene that was generated to be "7". After replace mutation, the chromosome would be, 1 2 3 7 5 6 7 8 9.

3. Swapping

In this type of mutation, first, two mutation points are selected randomly. The genes at these two points are then swapped to form the new chromosome.

Consider the following chromosome.

1 2 3 4 5 6 7 8 9

Assume that the random mutation points generated were 2 and 5. So, the chromosome becomes,

1 5 3 4 2 6 7 8 9

4. Scramble

In this type of mutation, first, two mutation points are selected randomly. The genes between these two points are then shuffled in random order to form the new chromosome.

Consider the following chromosome.

1 2 3 4 5 6 7 8 9

Assume that the random mutation points generated were 2 and 5. Assuming some random shuffling, the chromosome in after mutation would be,

1 3 5 4 2 6 7 8 9

Q. 8 How traditional algorithm differ from genetic algorithm ?

Ans. :

GA has the following significant differences over the traditional search techniques.

1. A GA works on coded versions of the problem parameters instead of the actual parameters themselves.
2. Unlike other conventional methods that search from a single point, a GA always operates on a whole population of points (strings). This makes the GA more robust.
3. Operating on the entire population also makes it possible to reach the global optimum and reduces the risk of becoming trapped in a local stationary point.
4. Usually genetic algorithms do not use any auxiliary information or about the objective function value such as derivatives. Hence, they can be applied to any kind of continuous or discrete optimization problems.
5. Conventional methods of search and optimization use deterministic transition operators while GA uses probabilistic transition operators.

Genetic algorithms may provide a number of potential solutions to a given problem and the choice of the final is left up to the user.

Q. 9 Explain steps involved in general genetic algorithm.

Ans. :

The first step in applying GA to a problem is the encoding scheme.GA requires that the actual parameter space be converted into genetic space. In general, the GA evolves a multi-set of chromosomes.

The chromosome is usually expressed as a string of variables, each element of which is called a gene. The variable can be represented as binary, real number, or other forms and its range is usually defined by the problem specified.

The steps involved in general GA are as follows.

1. Initialization of population

GA works on the population of chromosomes. So, the first step is to select the initial population. There are two parameters that need to be decided for initial population: size of the population and the method of selecting initial population. In general, the size of 20 to 50 is preferable. There are two ways to generate the initial population, namely, the random initialization and heuristic initialization.

2. Evaluation of Fitness function

A **fitness function** in a genetic algorithm is a function which is used to measure the "quality" or "fitness" of a given chromosome. Depending on the "fitness", chromosome either survives or dies in the process of evolution in a genetic algorithm. This is in sync with the Darwinian theory of survival of the fittest.

If a particular chromosome has a higher fitness value, it is more likely that it will survive and pass on to the next generation. On the other hand, chromosomes with lower fitness values have very low chances of survival. GAs, when used to solve optimization problems, derive the fitness function from the objective function. In general, the fitness function is always problem dependent.

For maximization problems, the fitness function 'F(x)' can be considered to be the same as the objective function 'f(x)' and hence $F(x) = f(x)$.

For minimization problems, the following transformation is often used.

$$F(x) = \frac{1}{1 + f(x)}$$

This transformation converts a minimization problem to an equivalent maximization problem.

3. **Generation of new population**

After fitness evaluation, a new population is generated. This new population is generated using three genetic operators: (a) Selection (b) Crossover and (c) Mutation.

(a) **Selection :** Selection is a mechanism for selecting individuals (strings) from a population according to their fitness (objective function value). The highest rank chromosome will have more possibility of selection and the worst will be eliminated.

(b) **Crossover :** Once the selection process is over, the crossover operator is applied next. Crossover is a recombination operator that combines subparts of two parent chromosomes to produce offspring that contains some parts of both parents' genetic material. In the crossover, highly fit individuals are given opportunities to reproduce by exchanging pieces of their genetic information with other highly fit individuals.

This produces new "offspring" solutions, which share some good characteristics taken from both parents.

(c) **Mutation :** The selection and crossover operators will generate a large amount of different off springs. However, there are two main problems with this.

(1) Depending upon the initial population chosen, there may not be enough diversity in the initial strings to ensure that the GA searches the entire problem space and

(2) The GA may converge on sub-optimum strings due to a bad choice of initial population.

These problems may be overcome by the introduction of mutation operator into GA. The mutation operator is used to inject new genetic material into the genetic population.

Mutation operator changes 1 as 0 and vice versa bit wise. Bitwise mutation is done bit by bit by flipping a coin with low probability. If the outcome is true, then the bit is altered; otherwise the bit is not altered.

Q. 10 Explain Holland's schema theorem.

Ans. :

Schema theorem serves as the analysis tool for the GA process.

1. Applicable to a canonical GA
2. binary representation
3. fixed length individuals
4. fitness proportional selection
5. single point crossover
6. gene-wise mutation

Schema

A schema is a set of binary strings that match the template for schema H. A template is made up of 1s, 0s, and *s where* is the 'don't care' symbol that matches either 0 or 1.

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 $2^l = 2^2$ different schemas

**, *0, 1*, 10

Schema : Order o(H)

The order of a schema is the number of its fixed bits, i.e. the number of bits that are not '*' in the schema H

Example : if H = 10*1* then o(H) = 3

Defining Length δ (H)

The defining length is the distance between its first and the last fixed bits

Example : if H = *1*01 then $\delta(H) = 5 - 2 = 3$

Example : if H = 0**** then $\delta(H) = 1 - 1 = 0$

Count

Suppose x is an individual that belongs to the schema H, then we say that x is **an instance of H ($x \in H$).**

$m(H, k)$ denotes the number of instances of H in the k th generation.

Fitness

$f(x)$ denotes fitness value of x.

$f(H, k)$ denotes **average fitness of** H in the k-th generation.

$$f(H, k) = \frac{\sum_{x \in H} f(x)}{m(H, k)}$$

Effect of GA on a Schema

Effect of Selection = Effect of Crossover + Effect of Mutation= Schema Theorem

Effect of Selection on Schema

Assume a GA with proportional selection and arbitrary but fixed fitness.

Selection probability for the individual x *is given as,*

$$p_s(x) = \frac{f(x)}{\sum_{i=1}^{N} f(x_i)}$$

Where, the N is the total number of individuals.

Net Effect of selection

The expected number of instances of H in the mating pool $M(H,k)$ is,

$$M(H, k) = \frac{\sum_{x \in H} f(x)}{\bar{f}} = m(H, k)\frac{f(H, k)}{\bar{f}}$$

"Schemas with fitness greater than the population average are likely to appear more in the next generation"

Effect of Crossover on Schema

Assume a single-point crossover.

Schema H survives crossover operation if,

1. one of the parents is an instance of the schema H **AND**
2. one of the offspring is an instance of the schema H

Crossover Survival examples

Consider H = *10**

$P_1 = 1\ 1\ 0 \mid 1\ 0 \in H$
$P_2 = 1\ 0\ 1 \mid 1\ 1 \notin H$
$\Rightarrow$
$S_1 = 1\ 1\ 0\ 1\ 1 \in H$ Schema H
$S_2 = 1\ 0\ 1\ 1\ 0 \notin H$ Survived

$P_1 = 1\ 1 \mid 0\ 1\ 0 \in H$
$P_2 = 1\ 0 \mid 1\ 1\ 1 \notin H$
$\Rightarrow$
$S_1 = 1\ 1\ 1\ 1\ 1 \notin H$ Schema H
$S_2 = 1\ 0\ 0\ 1\ 0 \notin H$ destroyed

Crossover Operation

Assuming a parent is an instance of a schema H. When the crossover is occurred within the bits of the defining length, it is destroyed unless the other parent repairs the destroyed portion.

Given a string with length l and a schema H with the defining length $\delta(H)$, the probability that the crossover occurs within the bits of the defining length is $\delta(H)/(l-1)$

Crossover Probability Example

Assuming H = *1**0

For,

$l = 5$

$\delta(H) = 5 - 2 = 3$

Thus, the probability that the crossover occurs within the defining length is ¾.

Crossover Operation

The upper bound of the probability that the schema H being destroyed is,

$$D_c(H) \leq p_c \frac{\delta(H)}{l-1}$$

Where, p_c is the crossover probability.

Net Effect of Crossover

The lower bound on the probability $S_c(H)$ that H survives is,

$$S_c(H) = 1 - D_c(H) \geq 1 - p_c \frac{\delta(H)}{l-1}$$

"Schemas with low order are more likely to survive"

Mutation Operation

Assumption : mutation is applied gene by gene.

For a schema H to survive, all fixed bits must remain unchanged.

Probability of a gene not being changed is $(1 - p_m)$.

Where , p_m is the mutation probability of a gene.

Net Effect of Mutation

The probability a schema H survives under Mutation

$$S_m(H) = (1 - p_m)^{o(H)}$$

"Schemas with low order are more likely to survive"

Putting it all together,

Schema Theorem

Expected number of Schema H in next generation >=

$$E[m(H, k+1)] \geq m(H, k)\frac{f(H, k)}{\bar{f}}\left(1 - p_c \frac{\delta(H)}{l-1}\right)(1 - p_m)^{o(H)}$$

"The schema theorem states that the schema with above average fitness, short defining length and lower order is more likely to survive"

Q. 11 Classify genetic algorithms

Ans. :

Genetic algorithms can be broadly classified as sequential GA and parallel GA.

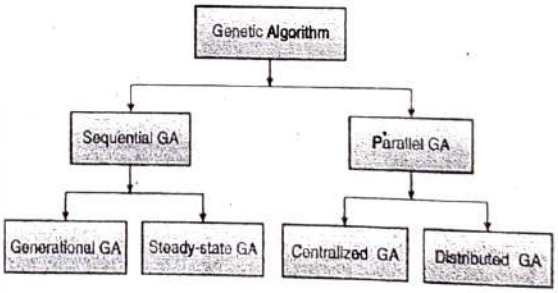


Fig. 5.11 : Classification of Genetic algorithms

Sequential GA

In sequential genetic algorithms, we proceed in an iterative manner, where, in each iteration, we generate a new population of strings from the old ones. Every string is the encoded version of a tentative solution. Every string's suitability to the problem is evaluated based on its fitness measure.

In order to compute a whole generation of new strings, the algorithm applies stochastic operators such as selection, crossover and mutation on an initial random population.

In sequential GA there are again two variants : steady state GA and generational GA.

The steady state GA is a simpler version of the generational GA. In steady state GA, the entire current population is not replaced, rather, two best parents are selected from the population and crossed obtaining two offspring that are then mutated and inserted in the current population.

On the other hand, in the generational version, a large portion of the population is selected and crossed (typically half the individuals), the resulting offspring is mutated and inserted into the population, thus replacing the old individuals.

Steady state GA is much simpler. In terms of computation, both versions are more or less equivalent, but the steady state GA requires more generations to converge, but its computational cost (i.e., the cost of every iteration) is much lower than the cost of the generational GA

Parallel GA

Parallel GAs (PGAs) are parallel versions of sequential GAs. The basic idea behind most parallel programs is to divide a task into chunks and to solve the chunks simultaneously using multiple processors. This divide-and-conquer approach can be applied to GAs. The members of the population can be partitioned into subpopulations that are distributed among different processors.

There are two classes of Parallel GAs: Centralised GA and distributed GA. Centralised GAs are also called **Global GA**. There is a single population (just as in simple GA), but the evaluation of fitness is distributed among several processors. Since, in this type of GA selection and crossover consider the entire population, it is also called giobal parallel GA.

The algorithm can be executed on a shared memory multiprocessor, where the chromosomes are stored in the shared memory and each processor only develops certain chromosomes.

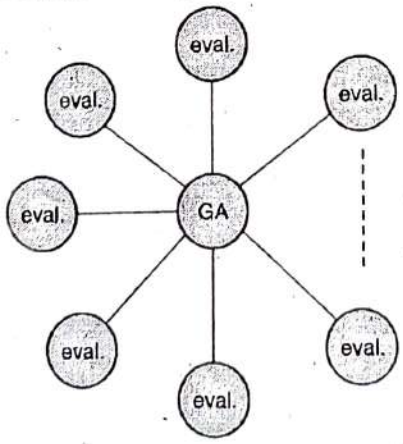


Fig. 5.12 : Schematic of centralised GA

copy of the reproduction. Distributed GA

as :

1. Single - population Fine-grained
2. Multiple-population Coarse-grained.

Q. 12 Explain the working of Holland's bucket brigade algorithm.

Ans.:

Bucket brigade algorithm evaluates rules and decides among competing alternatives. It Use a Genetic Algorithm (GA) to generate new rules. A classifier's strength U is used as its fitness. In each time step t, we assign a strength value ($G_{i,t}$) to each classifier R_i. This strength value represents the importance of the classifier. The strength value of each classifier is adapted depending upon the feedback from the environment (called payoff).

Each matching classifier computes a **'bid'** by multiplying its specificity (the number of non-don't cares in its condition part) with the product of its strength and a learning parameter.

The bid of the classifier R_i at t is defined as :

$$Bi, t = C_L G_{i,t} S_i$$

Where , C_L is a learning parameter, S_i is the specificity, $G_{i,t}$ is the strength of the classifier.

For small value of C_L, the system adapts slowly. If C_L high the strength tends to oscillate chaotically. Then the rules have to compete for the right for placing their output messages on the list. This can be done by random experiment like the selection in GA.

For each biding classifier it is decided randomly if it wins or not, where the probability that it wins is proportional to its bid. In this approach, more than one classifiers can win.

Another approach is **highest bidding** agent wins. This method resolves the conflict between two winning classifiers.

Payoff : each classifier receives a portion of Payoff from the environment and accordingly the strength of the classifier is adapted.

Denoting the set of classifiers, which have supplied the winning agent R_i, in step t with $S_{i,t}$ and then the new strength of a winning agent is reduced by its bid and increased by its portion of the payoff P_t received from the environment.

$$U_{i,t} + 1 = U_{i,t} + P_t / W_t - Bi, t$$

Where W_t is the number of winning agents in the actual time step. The winning agent pays its bid to the supplier which share the bid among each other equally.

If the winning agent is also active in the previous step and supplies another winning agent, the value above is additionally increased by one portion of the bid the consumer offers.

The strength of all other classifiers R_m, which are neither winning agents nor suppliers of winning agents are reduced by a certain factor.

$$U_{m,t+1} = U_{m,t}(1-T)$$

T is small value lying in the interval [0,1].

Thus we punish that classifier which never contributes anything in the output of the system.

The central idea is that classifiers which are not active when the environment gives payoff but which had an important role for setting the stage for directly rewarded classifiers can earn credit by participating in 'bucket brigade chains'.

Q. 13 Explain the working of Genetic programming with the help of flowchart.

Ans. :

Genetic programming typically starts with a population of randomly generated computer programs composed of the available programmatic ingredients.

The executional steps of genetic programming are as follows :

1. Randomly create an initial population (generation 0) of individual computer programs composed of the available functions and terminals.
2. Iteratively perform the following sub-steps (called a generation) on the population until the termination criterion is satisfied :
   (a) Execute each program in the population and ascertain its fitness (explicitly or implicitly) using the problem's fitness measure.
   (b) Select one or two individual program(s) from the population with a probability based on fitness (with reselection allowed) to participate in the genetic operations in (c).
   (c) Create new individual program(s) for the population by applying the following genetic operations with specified probabilities :
      (i) **Reproduction** : Copy the selected individual program to the new population.
      (ii) **Crossover** : Create new offspring program(s) for the new population by recombining randomly chosen parts from two selected programs.
      (iii) **Mutation** : Create one new offspring program for the new population by randomly mutating a randomly chosen part of one selected program.
      (iv) **Architecture-altering operations:** Choose an architecture-altering operation from the available repertoire of such operations and create one new offspring program for the new population by applying the chosen architecture-altering operation to one selected program.
3. After the termination criterion is satisfied, the single best program in the population produced during the run (the best-so-far individual) is harvested and designated as the result of the run. If the run is successful, the result may be a solution (or approximate solution) to the problem.

Fig. 5.13 is a flowchart showing the executional steps of a run of genetic programming. The flowchart shows the genetic operations of crossover, reproduction, and mutation as well as the architecture-altering operations. This flowchart shows a two-offspring version of the crossover operation.

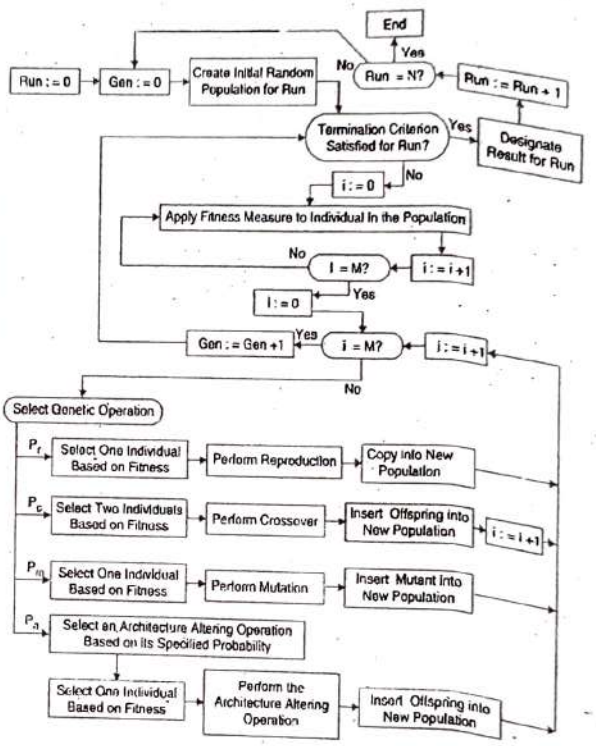


Fig. 5.13 : The flow chart of Genetic Programming

GP starts with an initial population of Computer programs composed of terminals appropriate to the problem. The individual program in the population is executed. Each individual program's performance (also called fitness) is compared.

The fitness of a computer program may be measured in many ways for example the time required to bring the system in desired target state, number of errors between the actual output and desired output, the accuracy of the program in recognizing a pattern etc.

For many problems, each program is executed over some representative test cases called fitness cases. This fitness cases may represent the different values for program's input, different initial condition or a different environment etc., The difference in the fitness is then exploited by GP.

GP applies Darwinian's selection and the genetic operations such as crossover, mutation, reproduction and architectural alteration to create a new population. These genetic operations are

applied to the individual that are probabilistically selected from the population based on the fitness. Here better individuals are favored.

After the genetic operations are performed on the current population. The new generation replaces the current population. This iterative process is repeated over many generations. The GP terminates when the termination condition is met.

Q. 14 What are the applications of genetic algorithm ?

Ans. : Applications of genetic algorithm

1. **Optimization**

GAs have been used in a wide variety of optimization tasks, including numerical optimization and combinatorial optimization problems such as Travelling Salesman Problem (TSP), circuit design, job scheduling, video and sound quality optimization etc.

2. **Automatic programming**

GAs have been extensively used to evolve computer programs for specific tasks and to design other computational structures, for example, cellular automata and sorting networks.

3. **Synthesis of Neural Network's architecture**

GAs can also be used to design neural networks. GAs can be used to optimize neural networks' structural parameters.

4. **Robotics**

GA techniques have been applied to the task of planning the path which a robot arm is to take in moving from one point to another. GAs can also be used to learn behaviour of the robot.

5. **Economic models**

GAs have been widely used to forecast financial markets, to develop bidding strategies and also to model processes of innovation.

6. **Control systems**

GAs are used in control systems engineering. GA can be applied to a number of control methodologies for the improvement of the overall system performance.

Q. 15 What are advantages and limitations of GA ?

Ans. : Advantages of GA

1. GAs are easy to understand since they do not demand the knowledge of complex mathematics.
2. Good for noisy environment.
3. Easy to discover global optimum.
4. Can work for wider solution space.

Limitations of GA

1. Not all optimization problems can be solved by means of a GA. Such problems are called variant problems. In some of the problems the fitness function which generates a block of chromosome is not known or poorly known blocks.
2. GA doesn't always guarantee to find a global optimum.
3. GAs are not suitable for real-time control systems since the difference between the shortest and longest optimization response time is much larger than with conventional gradient methods. This property of GA limits the GA's use in real-time systems.
4. Choosing various parameters such as the size of population, mutation rate, crossover rate, the selection method and deciding a fitness function has to be done meticulously.

Q. 16 Write a short note on : Advances in Genetic Algorithms

Ans. :

Many advancements in the field of genetic algorithm have been seen since its inception. Recent advancement of GA includes,

1. Constraint based GA

Constrained Optimization Problems are those optimization problems in which we have to maximize or minimize a given objective function value that is subject to certain constraints. Therefore, not all results in the solution space are feasible. In such a scenario, crossover and mutation operators might give us solutions which are infeasible. Therefore, additional mechanisms have to be employed in the GA when dealing with constrained Optimization Problems.

Some of the most common methods are :

1. Using penalty functions which reduces the fitness of infeasible solutions, preferably so that the fitness is reduced in proportion with the number of constraints violated or the distance from the feasible region.
2. Using repair functions which take an infeasible solution and modify it so that the violated constraints get satisfied.
3. Not allowing infeasible solutions to enter into the population at all.
4. Use a special representation or decoder functions that ensures feasibility of the solutions.

2. GA Based Machine Learning

Genetic Algorithms also find application in Machine Learning. Classifier systems are a form of genetics-based machine learning (GBML) system that are frequently used in the field of machine learning. GBML methods are a niche approach to machine learning.

There are two categories of GBML systems :

1. **The Pittsburg Approach :** In this approach, one chromosome encoded one solution, and so fitness is assigned to solutions.
2. **The Michigan Approach :** one solution is typically represented by many chromosomes and so fitness is assigned to partial solutions.

It should be kept in mind that the standard issue like crossover, mutation, Lamarckian or Darwinian, etc. are also present in the GBML systems.

3. Hybrid Genetic systems

Hybrid genetic algorithms have received significant interest in recent years and are being increasingly used to solve real-world problems. A genetic algorithm is able to incorporate other techniques within its framework to produce a hybrid that reaps the best from the combination. Genetic algorithms can be combined with other technologies like neural networks, fuzzy logic or any other intelligent systems.

4. Quantum Inspired GA

A novel evolutionary computing method called quantum genetic algorithms has emerged where principles of quantum mechanics are used to inspire more efficient evolutionary computing methods.

Q. 17 Maximize the function $f(x) = x^2$ when $x \in [0, 31]$. Show computation of minimum two generations.

Ans. :

Here binary unsigned integer of length 5 is used to represent variable x.

Starting with initial population size of 4.

Here initial population P_1 is chosen randomly.

For each string in initial population we compute fitness value $f(x) = x^2$.

Performing selection on initial population P_1. Here Roulette-wheel selection is used as a reproduction operator. The selection phase will decide which of the strings in initial population will take part in generation of new offspring.

The process is shown in the Table 5.1(a).

Table 5.1(a) : Initial population and selection phase

String No.	Initial population p_1	X	f(x)	$Fl/\sum f$	Expected count f/f	Actual count
1	01101	13	169	169/1170 = 0.14	169/293 = 0.58	1
2	11000	24	576	576/1170 = 0.49	576/293 = 1.97	2
3	01000	8	64	64/1170 = 0.06	64/293 = 0.22	0
4	10011	19	361	361/1170 = 0.31	361/293 = 1.23	1
Sum			1170 ($\sum f$)	1.00	4.00	
Average			293 (f)	0.25	1.00	
Maximum			576	0.49	1.07	

From the above table, it is clear that, the actual count of string 1 and 4 is 1. Hence one copy of both the strings will be taken in the mating pool. For string 2, actual count is 2, Hence two copies of string 2 will be taken in mating pool. Since string 3 has actual count 0, it is eliminated and will not participate in further population generation process.

Table 5.1(b) shows the process of generating next population.

Table 5.1(b) : New population after cross over on P_2

String No.	Population P_2 after selection (Mating pool)	Mate (selected randomly)	Crossover site(selected randomly)	New population P_3 (after crossover)	x	f(x)
1	0110|1	2	4	01100	12	144
2	1100|0	1	4	11001	25	625
3	11|000	4	2	11011	27	729
4	10|011	3	2	10000	16	256
				Sum		1754
				Average		439
				Maximum		729

In the Table 5.1(b), mating pool contains one copy of string 1 and 4 and two copies of string 2 from initial population P_1. Now we perform crossover operation on these strings. Crossover points and crossover mates have been selected randomly. In this example, string 1 is mated (i.e. crossed over) with string 2. Similarly string 3 is mated with string 4 and vice versa. Take this population P_3 as a current population and process it further for generating next generation.

Table 5.1(c) : Selection on population P_3

String No.	Population P_3	x	$F(x) = x^2$	$f_i/\Sigma f$	$f_i/\bar{f}$ expected count	Actual count
1	01100	12	144	144/1754 = 0.08	144/439 = 0.32	0
2	11001	25	625	625/1752 = 0.35	625/439 = 1.42	1
3	11011	27	729	729/1754 = 0.41	729/439 = 1.66	2
4	10000	16	256	256/1754 = 0.15	256/439 = 0.58	1
Sum			1754 (Σf)	1.00	4.00	
Average			439 (f)	0.25	1.00	
Maximum			729	0.41	1.66	

The fitness value f(x) is increased from 576 to 729.

Table 5.1(d) shows process of generating further population.

Table 5.1(d) : New population after crossover on P_4

String No.	Population P_4 after selection	Mate (random selection)	Crossover points (Random selection)	New population P_5 (after crossover)	x	f(x)
1	11001	3	3	11011	27	729
2	11011	4	2	11010	26	676
3	11011	1	3	11001	25	625
4	10000	2	2	10001	17	289
				Sum →		2319
				Average →		580
				Maximum →		729

Q. 18 Define crossover rate and mutation rate.

Ans. : Crossover Rate

The crossover rate is the probability of crossover denoted by P_c. The probability varies from 0 to 1. It is calculated as the ratio of the number of pairs to be crossed to some fixed population.

Mutation Rate

Mutation rate is the probability of mutation. Typically for a population size of 30 to 200, mutation rate ranges between 0.001 and 0.5.

Chapter 6 : Swarm Intelligence

Q. 1 Explain Particle Swarm Optimization .

Ans. :

Particle Swarm Optimization (PSO) is inspired from the nature, social behavior and dynamic movements with communication of insects, birds and fish. The idea of PSO Algorithms is to solve optimization problems defined over a large search space. Each point in the search space has an associated numerical value called the fitness value and the goal is to choose the best fitness value called the global optimization point.

For doing the same, PSO makes use of agents termed **"particles"**, which move step by step in search of the best solution (global optimum). Accordingly, the flying experience of itself along with the flying experience of the other particles is taken into consideration by each particle to adjust its own flying. The collection of flying particles is called a **"swarm"**.

Each particle of the swarm maintains a record of two things: its best solution, **personal best**, *pbest* and the best value among all the particles, **global best**, *gbest*.

Each particle adjusts its travelling speed dynamically corresponding to the flying experiences of itself and its colleagues. Each particle modifies its position according to: its current position, its current velocity, the distance between its current position and *pbest and* the distance between its current position and *gbest*.

Q. 2 What is the velocity update equation of PSO?

Ans. :

Number of particles is chosen usually between 10 and 50.

c_1 is the importance of personal best value.

c_2 is the importance of neighborhood best value.

Usually $c_1 + c_2 = 4$ (empirically chosen value)

Particle's velocity is given by :

$$v_i^{t+1} = v_i^t + c_1 U_1^t (pb_i^t - p_i^t) + c_2 U_2^t (gb^t - p_i^t)$$

Diversification — intensification

1. **Intensification :** explores the previous solutions, finds the best solution of a given region.
2. **Diversification :** searches new solutions, finds the regions with potentially the best solutions.

If velocity is too low, then the algorithm becomes too slow and if velocity is too high, the algorithm becomes too unstable.

Q. 3 Write short note on : Premature Convergence of PSO

Ans. :

Although the speed of convergence is very fast, many experiments have shown that once PSO traps into local optimum, it is difficult for PSO to jump out of the local optimum. This leads to premature convergence of PSO. The lack of population diversity in PSO is understood to be a factor in its convergence on local optima. Therefore, the addition of a mutation operator to PSO should enhance its global search capacity and thus improve its performance.

There are different ways to avoid the premature convergence of PSO. Using a quantum model (QPSO), the traditional position and velocity equations are replaced with a wave function which can give optimal solutions. The gbest (the global best particle) and *mbest* (the mean value of all particles' previous best position) can be mutated with Cauchy distribution. This happens because the expectation of Cauchy distribution does not exist and so the variance of Cauchy distribution is infinite. This helps to introduce diversification in the PSO and thus prevent premature convergence.

Q. 4 What are the prevalent topologies of PSO? Explain in short.

Ans. :

The neighborhood of each particle can be decided depending upon the chosen topology to pass on the information. Popular topologies include the ring and adaptive random topologies.

1. The Ring Topology

The ring topology is a very common topology used for years because of its simplicity.

As per this topology, the neighborhood of a particle i is given by:

$$i - 1 \ mod(S), i, i + 1 \ mod(S)$$

For example, if S = 30 the neighbourhood of the particle 0 is {29, 0, 1}, and the neighbourhood of the particle 29 is {28, 29, 0}.

2. The Adaptive Random Topology

At the very beginning and after each unsuccessful iteration (no improvement of the best known fitness value), the graph of the information links is modified: each particle informs at random k particles (the same particle may be chosen several times), and informs itself. The parameter k is usually set to 3. It means that each particle informs at least one particle (itself), and at most k + 1 particles (including itself). It also means that each particle can be informed by any number of particles between 1 and S.

Q. 5 What are real valued and binary PSO ?

Ans. :

In regular (real valued) PSO, everything is in terms of a velocity. Generally the velocity is defined in terms of a probability of the bit changing. In Binary PSO, 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.

In BPSO, bit-by-bit updates are done probabilistically. In other words, for a chosen bit b in a chosen string s 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. (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.

Q. 6 Explain the Ant Colony Optimisation Algorithm with the help of an example.

Ans. :

Ant Colony Optimisation is a technique to solve optimisation problems based on the way that ants indirectly communicate directions to each other. The main idea is inspired by the natural behavior of ants when they set out in search of food.

The ants find the shortest paths between their nest and the food by means of a chemical substance they secret called the **pheromone.** This pheromone acts as a signal to other ants. If an ant decides with some probability to follow the pheromone trail, it itself lays more pheromone, thus reinforcing the trail. If more number of ants follows the trail, the pheromone becomes stronger and ants are more likely to follow it.

Q. 7 What is the Global Pheromone Update Rule and Transition Rule with respect to ACO?

Ans. :

Ants are *agents* that move along between nodes in a graph. They choose where to go based on pheromone strength (and may be other things)

An ant's path represents a specific candidate solution. When an ant has finished a solution, pheromone is laid on its path, according to quality of solution.

This pheromone trail affects behaviour of other ants. An ACO algorithm can be used typically to solve graph based problems like the Travelling Salesperson Problem (TSP). Consider, for example, a 4-city TSP instance.

Initially, random levels of pheromone are scattered on the edges.

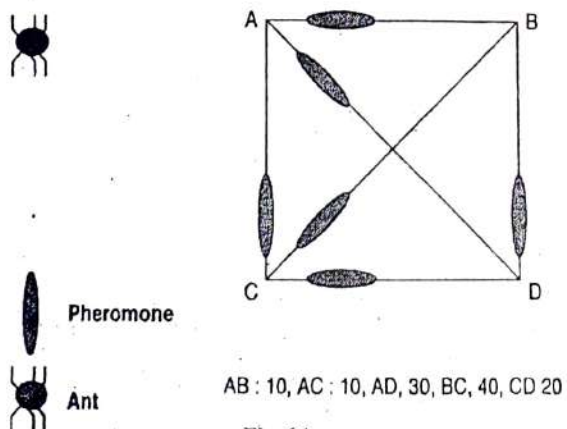


Fig. 6.1

An ant is placed at a random node.

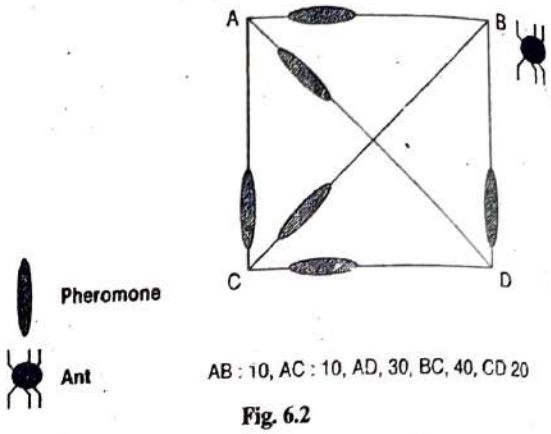


Fig. 6.2

The ant decides where to go from that node, based on probabilities calculated from :
- pheromone strengths,
- next-hop distances.

Suppose this one chooses BC

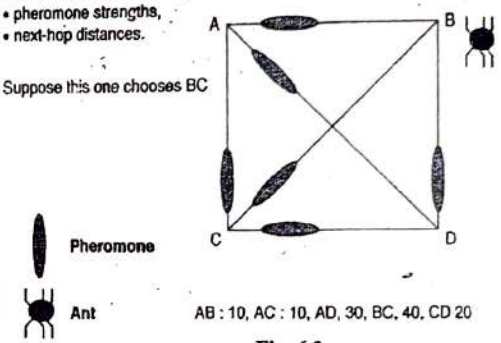


Fig. 6.3

The ant is now at C, and has a 'tour memory' = {B, C} – so he cannot visit B or C again.
Again, he decides next hop (from those allowed) based on pheromone strength and distance ;
Suppose he chooses CD.

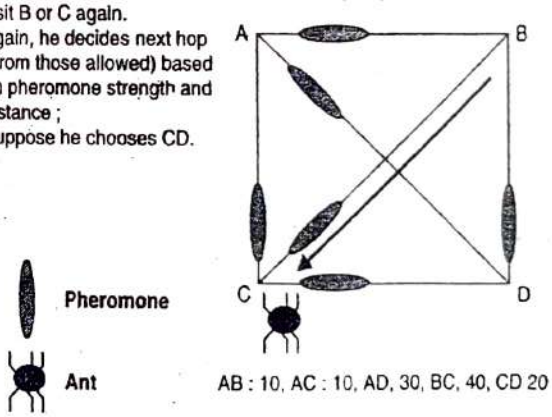


Fig. 6.4

The ant is now at D, and has a 'tour memory' = {B, C, D}
There is only one place he can go now :

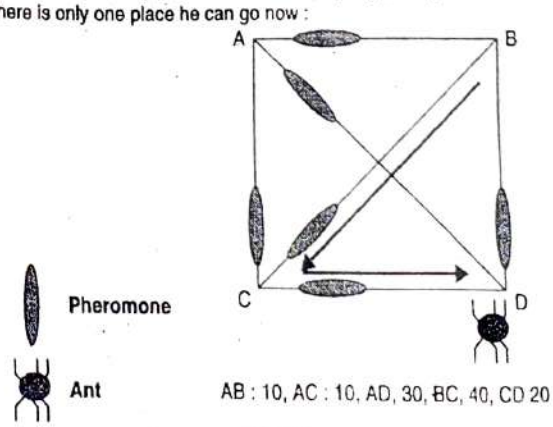
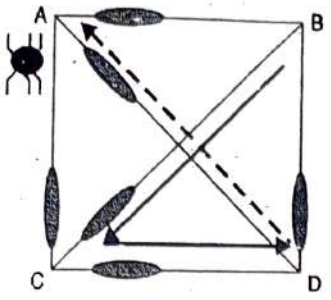


Fig. 6.5

So, he has nearly finished his tour, having gone over the links : BC, CD and DA.



Pheromone

Ant

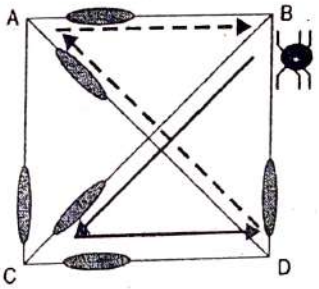
AB : 10, AC : 10, AD, 30, BC, 40, CD 20

Fig. 6.6

So, he has nearly finished his tour, having gone over the links : BC, CD, and DA.

AB is added to complete the round trip.

Now, pheromone on the tour is increased, in line with the fitness of that tour.



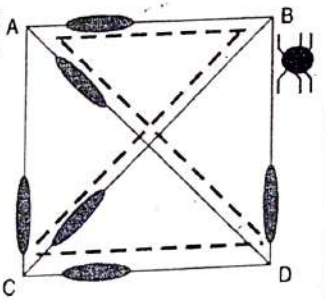
Pheromone

Ant

AB : 10, AC : 10, AD, 30, BC, 40, CD 20

Fig. 6.7

Next, pheromone everywhere is decreased a little, to model decay of trail strength over time.



Pheromone

Ant

AB : 10, AC : 10, AD, 30, BC, 40, CD 20

Fig. 6.8

We repeat with the same procedure by placing an ant at another random location.

The following is the ACO algorithm applied to TSP.

We have a TSP, with n cities.

1. **We place some ants at each city. Each ant then does this :** It makes a complete tour of the cities, coming back to its starting city, using a *transition rule* to decide which links to follow. By this rule, it chooses each next-city at random, but biased partly by the pheromone levels existing at each path, and biased partly by *heuristic information.*
2. **When all ants have completed their tours : Global** Pheromone Updating occurs. The current pheromone levels on all links are reduced (i.e. pheromone levels decay over time). Pheromone is lain (belatedly) by each ant as follows: it places pheromone on all links of its tour, with strength depending on how good the tour was.

Then we go back to 1 and repeat the whole process many times, until we reach a termination criterion.

The transition rule

$T(r, s)$ is the amount of pheromone currently on the path that goes directly from city r to city s. $H(r, s)$ is the heuristic value of this link - in the classic TSP application, this is chosen to be 1/distance(r, s) i.e. the shorter the distance, the higher the heuristic value. $p_k(r, s)$is the probability that ant k will choose the link that goes from r to s is a parameter that we can call the *heuristic strength.*

The rule is : $$p_k(r, s) = \frac{T(r, s) \cdot H(r, s)^{\beta}}{\sum_{\text{unvisited cities c}} T(r, c) \cdot H(r, c)^{\beta}}$$

Where our ant is at city r and s is a city as yet unvisited on its tour, and the summation is over all of k's unvisited cities.

Global pheromone update

$A_k(r, s)$ is amount of pheromone added to the (r, s) link by ant k.

m is the number of ants.

ρ is a parameter called the pheromone decay rate.

L_k is the length of the tour completed by ant k.

$T(r, s)$ at the next iteration becomes :

$$\rho \cdot T(r, s) + \sum_{k=1}^{m} + A_k(r, s)$$

Where, $$A_k(r, s) = \frac{1}{L_k}$$

Q. 8 Write the pseudo code for Ant Colony Optimisation.

Ans. :

Algorithm : The framework of a basic ACO algorithm

Input : An instance P of a CO problem model $P = (S, f, \Omega)$.

```
InitializePheromoneValues(T)
S_bs = NULL
while termination conditions not met do
  S_iter = ∅
  for j = 1, ..., n_a do
    S = ConstructSolution(T)
    if S is a valid solution then
      S = LocalSearch(S)   {optional}
      if (f(S) < f(S_bs)) or (S_bs == NULL) then S_bs ← S
      S_iter ← S_iter ∪ {S}
    end if
  end for
  ApplyPheromoneUpdate(T, S_iter, S_bs)
end while
```

Output : The best-so-far solution S_{bs}

Q. 9 What are the applications of PSO and ACO?

Ans. :

PSO is used in most optimization problems.

ACO is mostly used for graph based problems :

1. Travelling Salesman Problem
2. Quadratic Assignment Problem
3. Network Model Problem
4. Vehicle routing

❑❑❑