KTU LECTURE NOTES

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

DOWNLOAD KTU ASSIST APP FROM PLAYSTORE

GENETIC ALGORITHM

LECTURE 8

November 26, 2017

Introduction

- In *The Origin of Species*, Charles Darwin stated the theory f natural evolution.
- Over many generations, biological organisms evolve according to the principle of natural selection.
- GAs are powerful adaptive method to solve search and optimization problem.

What are Genetic Algorithms?

- *GAs* are adaptive heuristics search algorithm based on the evolutionary ideas of natural selection and genetics.
- Search−based optimization technique based on principles of *genetics and natural selection*.
- Used to find optimal solutions to difficult problems which otherwise would take a lifetime to solve.
- The main idea is *survival of the fittest*.

Why Genetic Algorithms?

- They are better than conventional algorithms.
- They are more robust.
- They do not break easily even if the inputs are changed slightly or in the presence of reasonable noise.

Comparison

Table 1.1: Classical Algorithm VS Genetic Algorithm

Classical Algorithm	Genetic Algorithm
Generates a single point at each iteration. The sequence of points approaches an optimal solution.	Generates a population of points at each iteration. The point in the population approaches an optimal solution.
Select the next point in the sequence by a deterministic computation.	Select the next population by computation which uses random number generators.

Biological background

- The science that deals with the mechanisms responsible for similarities and differences in a species is called *genetics*.
- The word *genetics* is derived from the Greek word *genesis* meaning *to grow* or *to become*.
- It helps us to differentiate between heredity and variations during the process of evolution.

Terminologies

- *The Cell*
  - Every animal/human *cell* is a complex of many small factories tha work together.
  - Center of this is *cell nucleus*.
  - The genetic information is contained in the cell nucleus.

- *Chromosomes*
  - All the genetic information gets stored in the *chromosomes*.
  - Each chromosome is build of *DNA*.
  - In humans, chromosomes exist in pairs.
  - Chromosomes are divided into several parts called *genes*.
  - Genes code the properties of species.
  - The combination of genes for one property are called *alleles*.

- *Chromosomes*(Contd...)
  - Set of all possible alleles present in a particular population forms *gene pool*.
  - Size of gene pool determines the diversity of the individual in th population.
  - Set of all the genes of a specific species is called *genome*.
  - Each and every gene has a unique position on the genome calle *locus*.
  - Entire combination of genes is called *genotype*.

Terminology relationships between natural evolution and genetic algorithm

Natural evolution	Genetic algorithm
Chromosomes	String
Gene	Feature or Character
Allele	Feature value
Locus	String position
Genotype	Structure or Coded string
Phenotype	Parameter set

Operators in Genetic Algorithm

1. *Encoding*
2. *Selection*
3. *Crossover*
4. *Mutation*

Encoding

- *Encoding* is a process of representing individual genes.
- The process can be performed using *bits*, *numbers*, *trees*, *arrays*, *lists* ⋯.
- It depends mainly on solving the problem.

Encoding Methods

1. *Binary Encoding*
2. *Octal Encoding*
3. *Hexadecimal Encoding*
4. *Permutation Encoding*
5. *Value Encoding*
6. *Tree Encoding*

Binary Encoding

- The most common way of encoding is a *binary* (*bit*) *string*.
- Binary coded strings with *1*s and *0*s.
- Each bit in the string can represent some characteristics of the solution.
- Every bit string therefore is a solution but not necessarily the best solution.
- The whole string can represent a number.
- The way bit strings can code differs from problem to problem.

- *Binary encoding* gives many possible chromosomes with a smaller number of *alleles*.
- The length of the string depends on the accuracy.

1. Integers are represented exactly.
2. Finite number of real numbers can be represented.
3. Number of real numbers represented with string length.

Chromosome 1	1 1 0 1 0 0 0 1 1 0 1 0
Chromosome 2	0 1 1 1 1 1 1 1 1 1 0 0

Octal Encoding

- *Octal encoding* uses string made up of *octal numbers* (*0–7*).

Chromosome 1	**03467216**
Chromosome 2	**15723314**

Hexadecimal Encoding

- *Hexadecimal encoding* uses string made up of *hexadecimal numbers* ($0-9$, $A-F$).

Chromosome 1	**9CE7**
Chromosome 2	**3DBA**

Permutation Encoding (Real Number Coding)

- Every chromosome is a string of numbers, represented in a sequence.
- In *Permutation encoding*, every chromosome is a string of *integer/real values*, which represent number in a sequence.
- It is useful for ordering problems.

Chromosome A	**1 5 3 2 6 4 7 9 8**
Chromosome B	**8 5 6 7 2 3 1 4 9**

Value Encoding

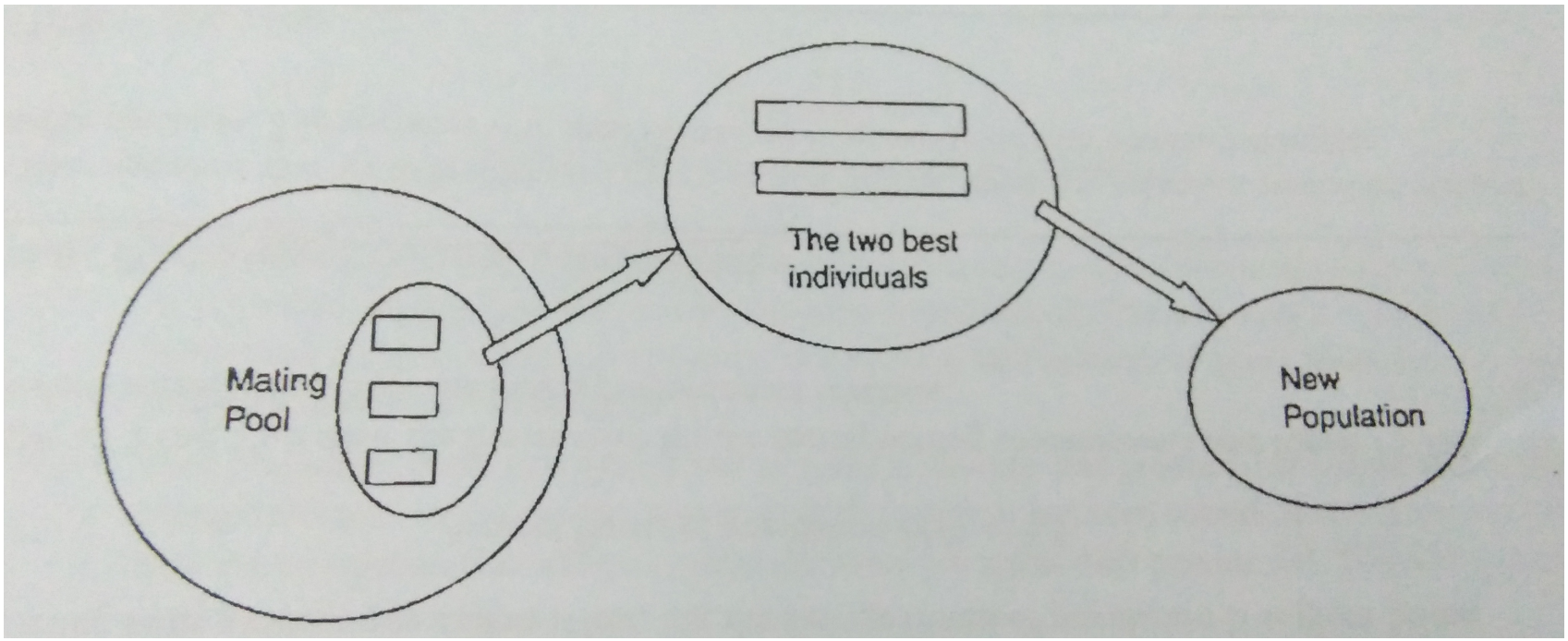
- In *Value encoding*, every chromosome is a string of some *values*.
- Values can be anything connected to problem, *numbers*, *real numbers* or *characters*.
- Value encoding is very good for some special problems.

Chromosome A	**1.2324 5.3243 0.4556 2.3293 2.4545**
Chromosome B	**ABDJEIFJDHDIERJFDLDFLFEGT**
Chromosome C	(*back*), (*back*), (*right*), (*forward*), (*left*)

Tree Encoding

- *Tree encoding* is mainly used for evolving program expressions for genetic programming.
- Every chromosome is a *tree* of some objects such as *functions* and *commands* of a programming language.

Selection



- *Selection* is a method that randomly picks chromosomes out of the population according to their evaluation function.
- The *selection pressure* is defined as the degree to which the better individuals are favored.
- Convergence rate of the *GA* is largely determined by magnitude of the selection pressure.

Types of Selection scheme

1. *Proportionate–based selection* picks out individuals based upon their fitness values relative to the fitness of the other individuals in the population.
2. *Ordinal–based selection* select individuals not upon their raw fitness, but upon their rank within the population.

Selection Methods

1. *Roulette Wheel Selection*
2. *Random Selection*
3. *Rank Selection*
4. *Tournament Selection*
5. *Boltzmann Selection*
6. *Stochastic Universal Sampling*

Roulette Wheel Selection

- The commonly used reproduction operator is the *proportionate reproductive operator*.
- A string is selected from the mating pool with a probability proportional to the fitness.
- The principle of *Roulette Selection* is a linear search through a *Roulette wheel* with the slots in the wheel weighted in proportion to the individual's fitness values.
- A target value is set which is a random proportion to the sum of the fitness in the population.
- The population is stepped through until the target value is reached.

1. *Sum the total expected value of the individuals in the population, T.*
2. *Repeat N times :*
    1. *Choose a random integer r between 0 and T.*
    2. *Loop through the individuals in the population, summing the expected values, until the sum is greater than or equal to r.*

       The individual whose expected value puts the sum over this limit is the one selected.

Random Selection

- *Random selection* randomly selects a parent from the population.

Rank Selection

- *Rank selection* ranks the population and every chromosomes receives fitness from the ranking.
- The worst has fitness *1* and the best has fitness *N*.
- It result in slow convergence but prevents too quick convergence.
- It also keeps up selection pressure when the fitness variance is low.
- It preserves diversity and hence leads to successful search.

1. *Select a pair of individuals at random.*
   Generate a random number R between 0 and 1.
   If $R < r$, use the first individual as parent.
   If $R \geq r$, then use the second individual as the parent.
   This is repeated to select the second parent.

2. *Select two individuals at random.*
   The individual with the highest evaluation becomes the parent.
   Repeat to find the second parent.

Tournament Selection

- *Tournament Selection* strategy provides selective pressure by holding a tournament competition among N_u individuals.
- The best individual from the tournament is the one with the highest fitness, who is the winner of N_u.
- Tournament competitions and the winner are then inserted into the mating pool.
- The tournament competition is repeated until the mating pool for generating new offspring is filled.
- This method is more efficient and leads to an optimal solution.

Boltzmann Selection

- In *Boltzmann Selection* a continuously varying temperature controls the rate of selection according to a present schedule.
- The temperature starts out high, which mens that the selection pressure is low.
- The probability that the best string is selected and introduced into the mating pool is very high.
- *Elitism* can be used to eliminate the chance of any undesired loss of information during the mutation stage.

Let f_{max} be the fitness of the currently available best string.
If the next string has fitness $f(X_i)$ such that $f(X_i) > f_{max}$, then the new string is selected.

Otherwise, it is selected with Boltzmann probability,

$$P = \exp[-[f_{max} - f(X_i)]/T]$$

where,

$$T = T_0(1 - \alpha^k) \text{ and } k = (1 + 100 * g/G)$$

g is the current generation number and G is the maximum value of g.
The value of α can be chosen from the range $[0, 1]$.
T_0 from range $[5, 100]$.

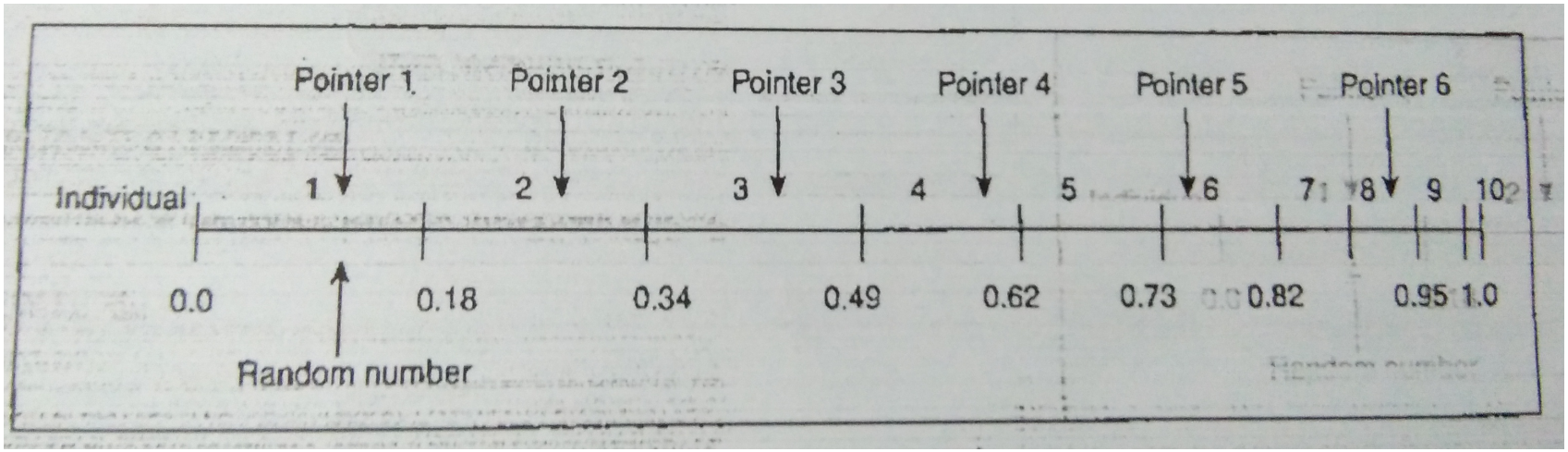
Stochastic Universal Sampling

- *Stochastic Universal Sampling* provides zero bias and minimum spread
- The individuals are mapped to contiguous segments of a line such that each individual's segment is equal in size to its fitness.
- Here, equally spaced pointers are placed over the line, as many as there are individuals to be selected.

Consider,
NPointer – The number of individuals to be selected.

Then,
distance between the pointers are, $\frac{1}{NPointer}$ *and*

the position of the first pointer is given by a randomly generated number in the range $[0, \frac{1}{NPointer}]$.



Crossover(Recombination)

- *Crossover* is the process of taking two parent solutions and producing from them a child.
- After the selection process, the population is enriched with better individuals.
- Reproduction makes clones of good strings but does not create new ones.

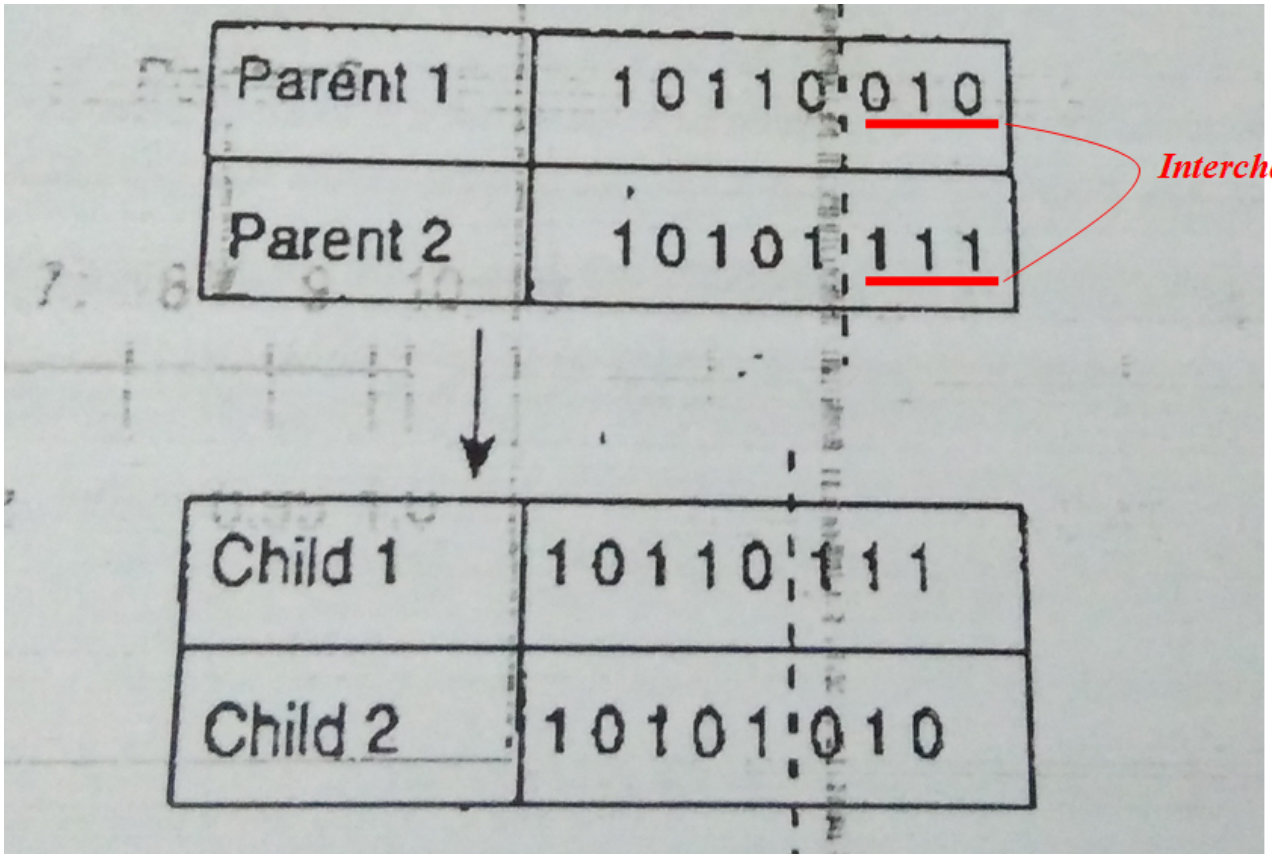
- *Crossover* is a recombination operator that proceeds in *three* steps:
  1. The reproduction operator selects at random, a pair of two individual string for the mating.
  2. A cross site is selected at random along the string length.
  3. Finally, the position values are swapped between the two strings following the cross site.

Crossover Techniques

1. *Single–Point Crossover*
2. *Two–Point Crossover*
3. *Multi–point Crossover*
4. *Uniform Crossover*
5. *Three–Parent Crossover*
6. *Crossover with Reduced Surrogate*
7. *Shuffle Crossover*
8. *Precedence Preservative Crossover*
9. *Ordered Crossover*
10. *Partially Matched Crossover*
11. *Crossover Probability*

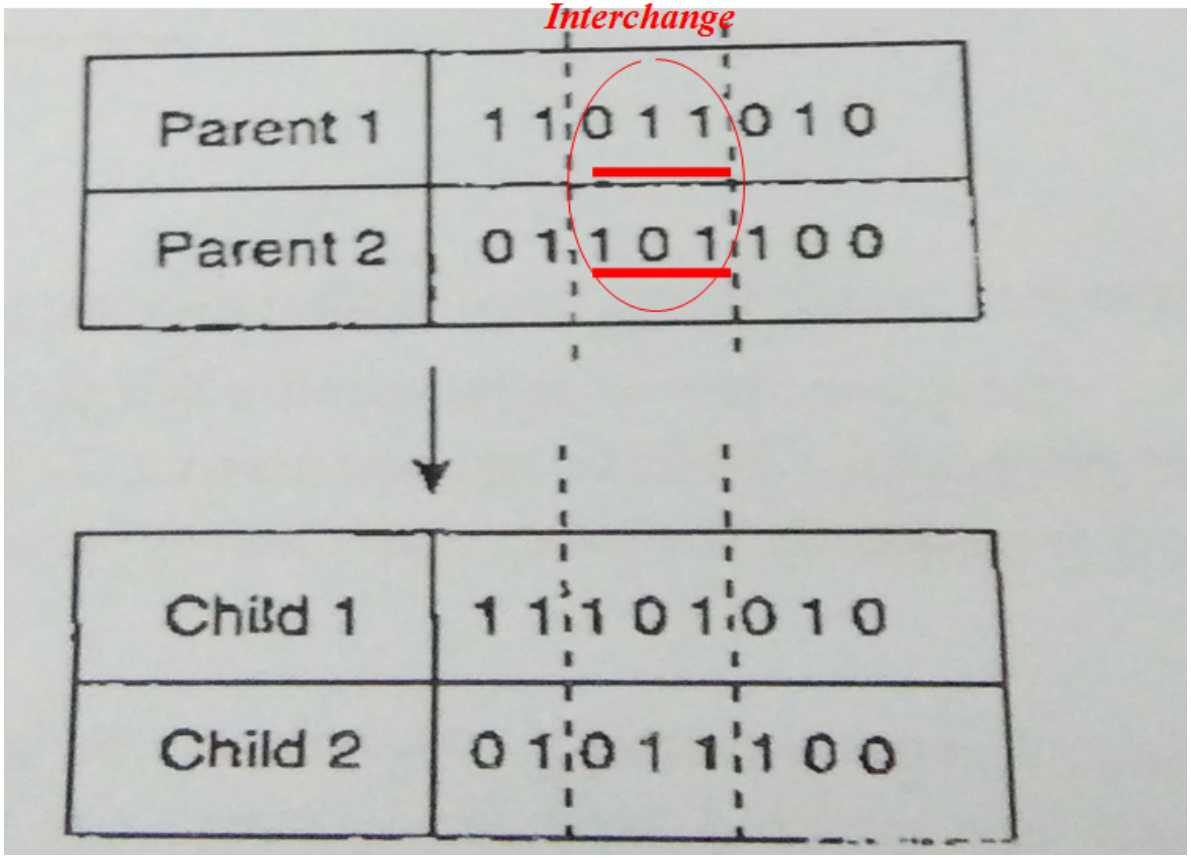
Single–Point Crossover

- The traditional *GA* uses *single–point crossover*, where the two mating chromosomes are cut once at corresponding points and the sections after the cuts are exchanged.
- A *cross site or cross point* is selected randomly along the length of the mated strings and bits next to the cross sites are exchanged.



Two−Point Crossover

- Involving more than one cut point.
- In *two−point crossover*, two crossover points are chosen and the contents between these pointers are exchanged between two mated parents.
- Adding further crossover points reduces the performance of *GA*.



Multi−Point Crossover

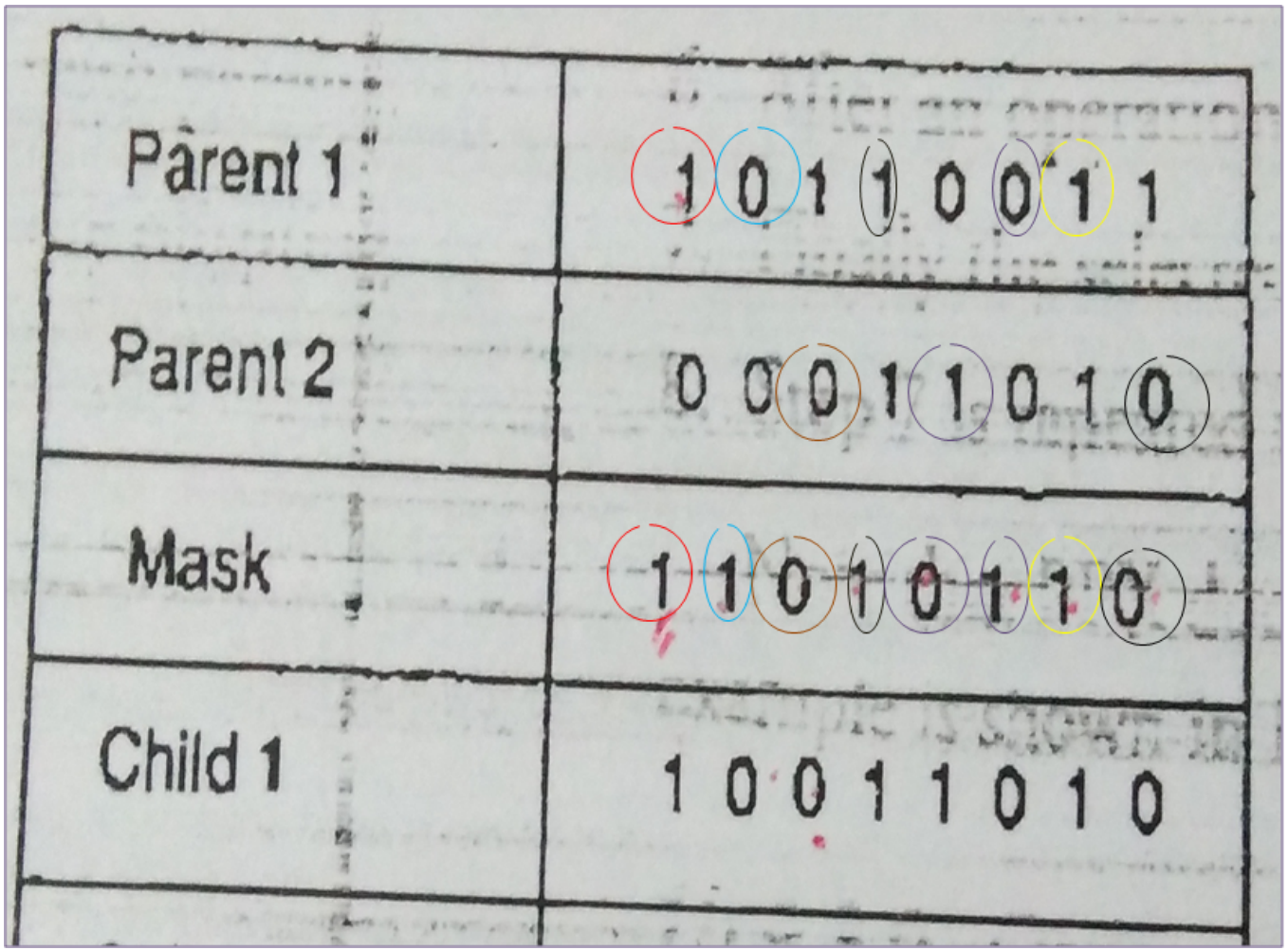
- Two ways:
  1. *Even number of cross sites* The cross sites are selected randomly around a circle and information is exchanged.
  2. *Odd number of cross sites* A different cross point is always assumed at the string beginning.

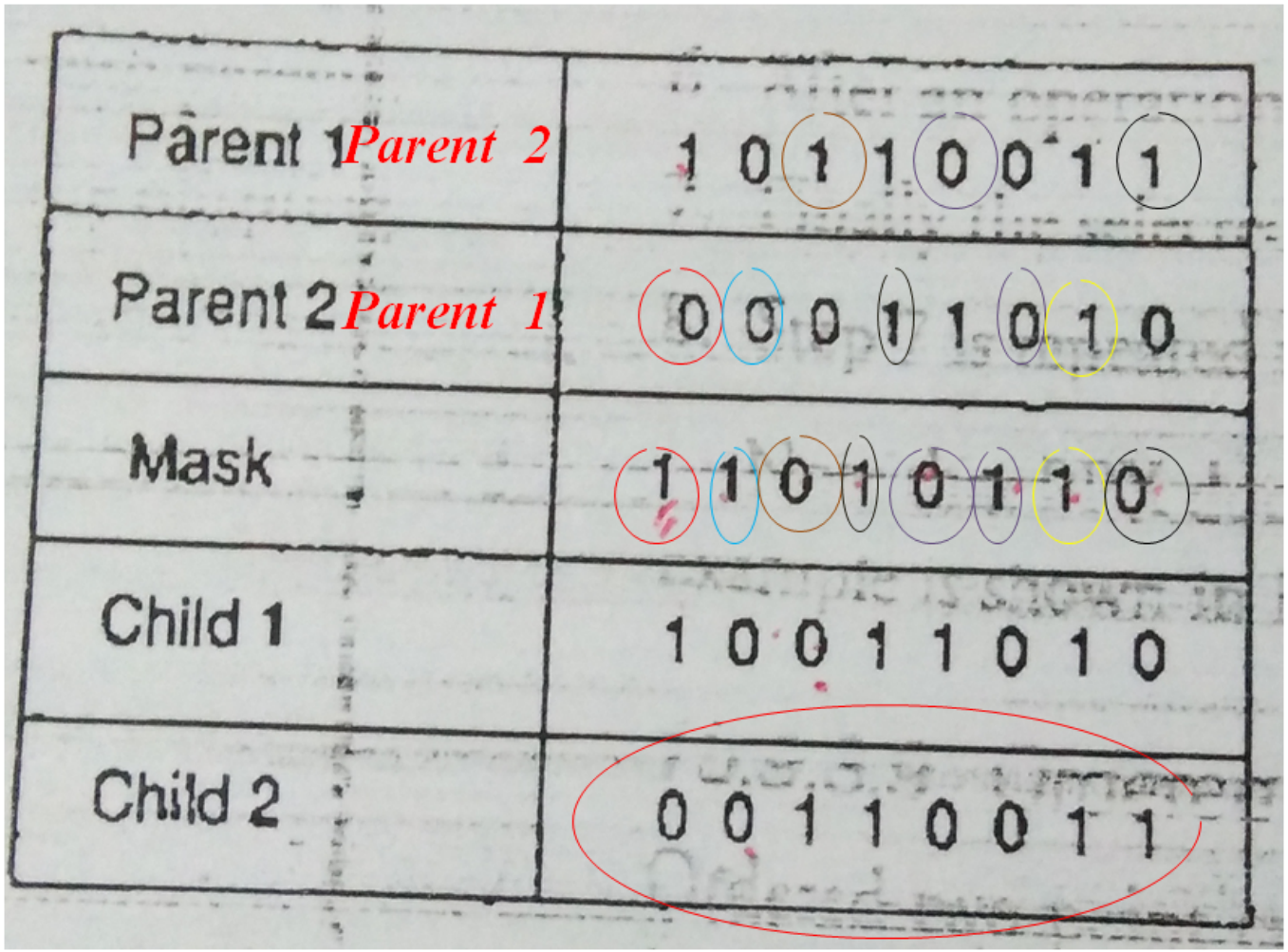
Uniform Crossover

- Each gene in offspring is created by copying corresponding gene from one or other parent chosen according to a random generated *binary crossover mask* of the same length as the chromosomes.
- When there is a *1* in the crossover mask, the gene is copied from the first parent and where there is a *0* in the mask, the gene is copied from the second parent.
- A new crossover mask is randomly generated for each pair of parents.
- Number of crossing point is,

$$\frac{L}{2}$$

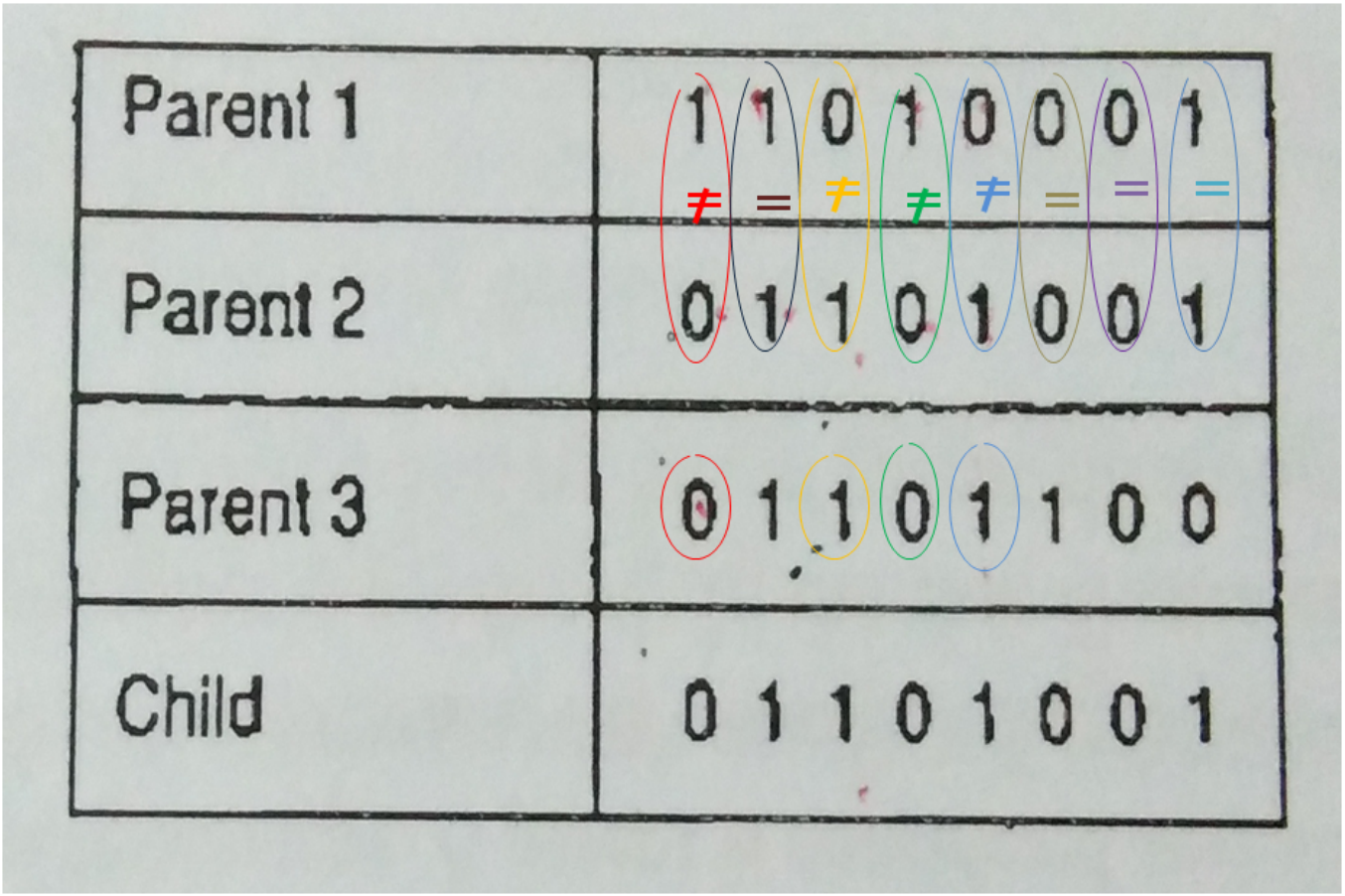
where L is the chromosome length.





Three−Parent Crossover

- Three parents are randomly chosen.
- Each bit of the first parent is compared with the bit of the second parent.
- If both are the same, the bit is taken for the offspring, otherwise the bit from the third parent is taken for the offspring.



Crossover with Reduced Surrogate

- The *reduced surrogate operator constraints crossover* to always produce new individuals wherever possible.
- Implemented by restricting the location of crossover points such that the crossover points only occur where gene values differ.

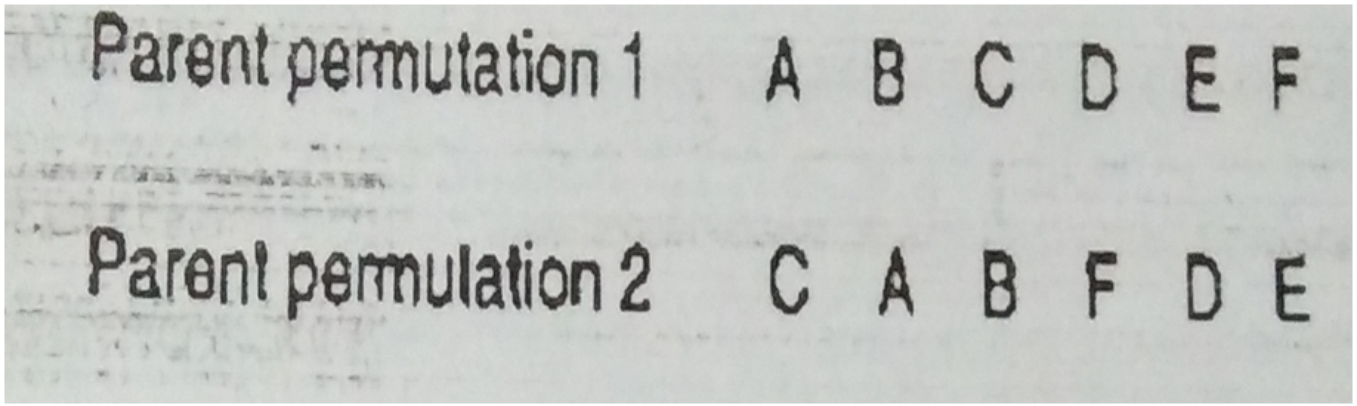
Shuffle Crossover

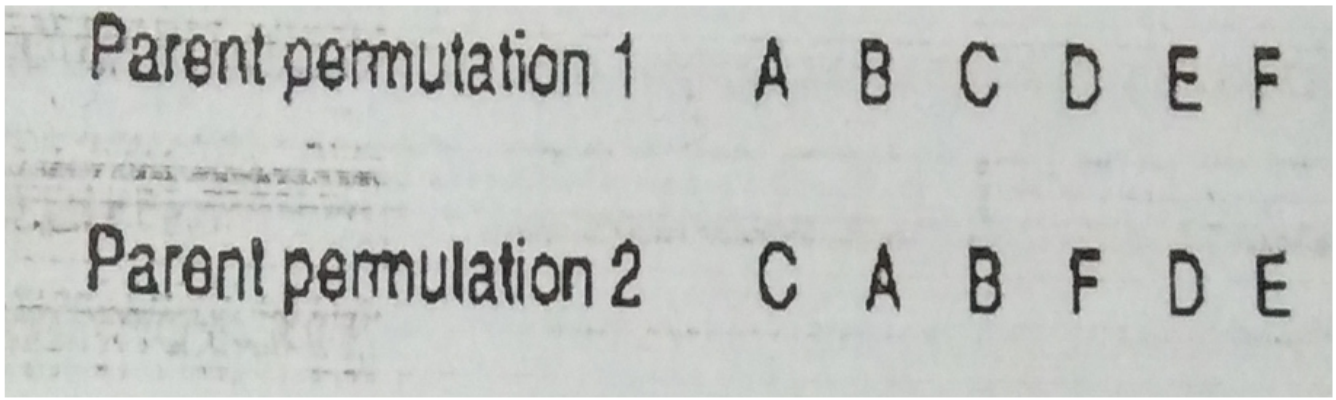
- *Shuffle crossover* is related to *uniform crossover*.
- A single crossover position is selected.
- Before the variables are exchanged, they are randomly shuffled in both parents.
- After recombination, the variables in offspring are shuffled

Precedence Preservative Crossover

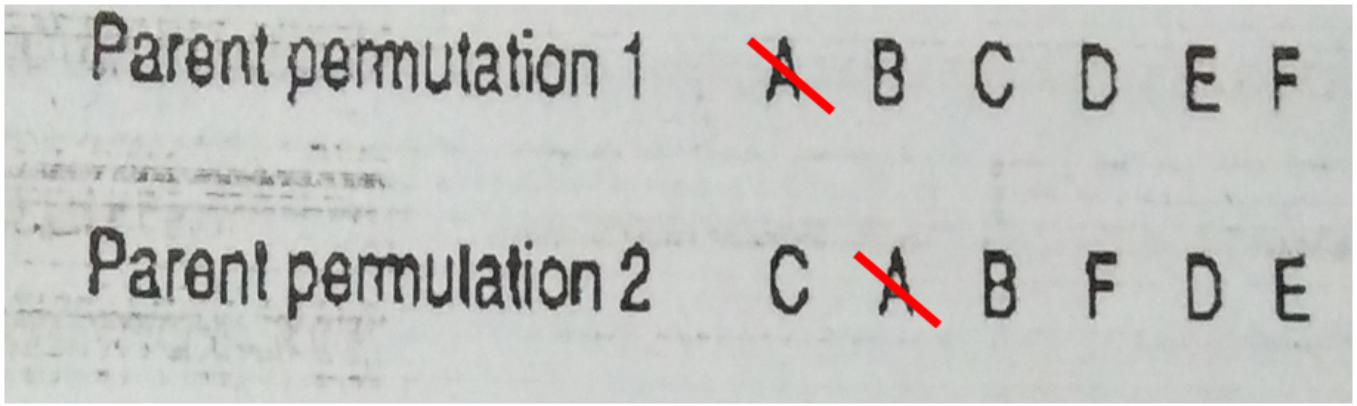
- The operator passes on precedence relations of operations given in two parental permutations to one offspring at the same rate, while no precedence relations are introduced.
- PPX is introduced for a problem consisting of six operations $A-F$.
- The operator works as follows:
  - *A vector of length Sigma, sub $i = 1$ to m_i, representing the number of operations involved in the problem, is randomly filled with elements of the set $[1, 2]$.*
  - *This vector defines order in which operations are successively drawn from parent 1 and parent 2.*

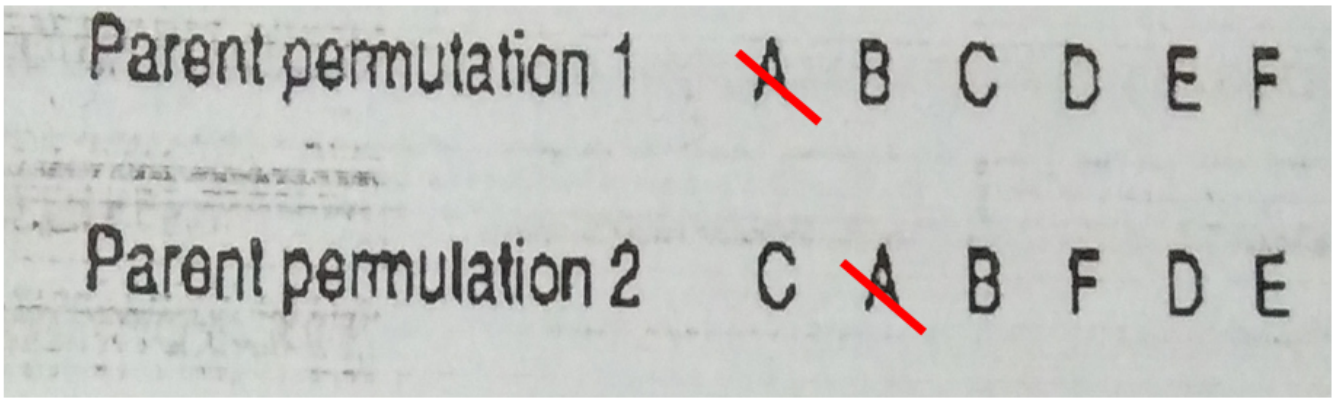
- *Consider the parent and offspring permutations as lists, for which the operations " append " and " delete " are defined.*
- *First step is to initialize an empty offspring.*
- *The leftmost operations in one of the two parents is selected in accordance with the order of parents given in the vector.*
- *After an operation is selected, it is deleted in both parents.*
- *Finally, the selected operation is appended to the offspring.*
- *Process is repeated until both parents are empty and the offspring contains all operations involved.*



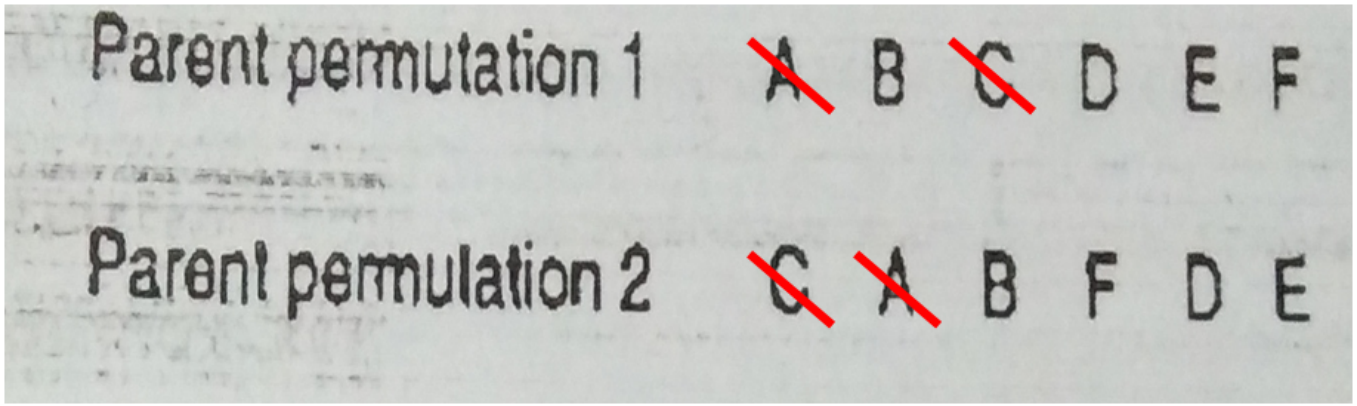


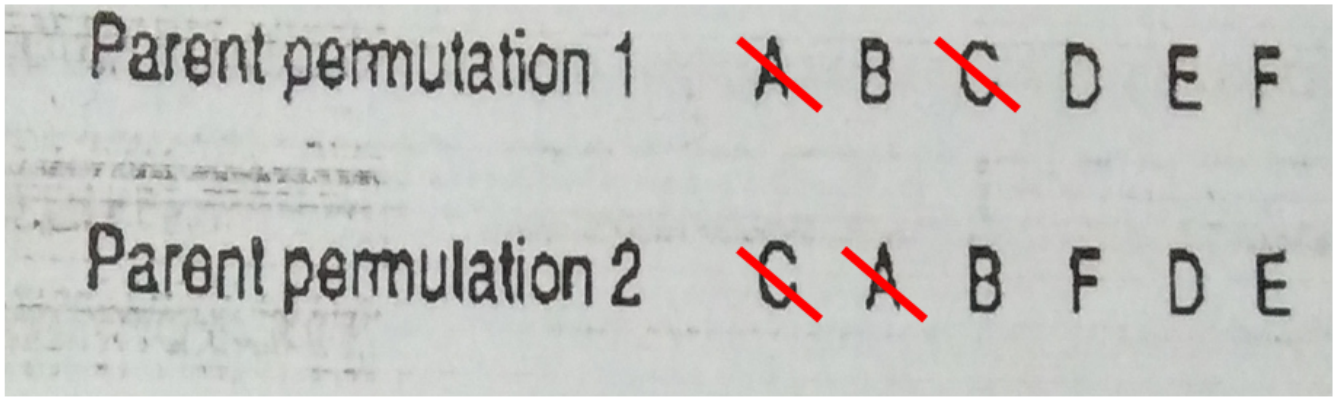
Select Parent No. (1/2)	*1*					
Offspring Permutation	*A*					



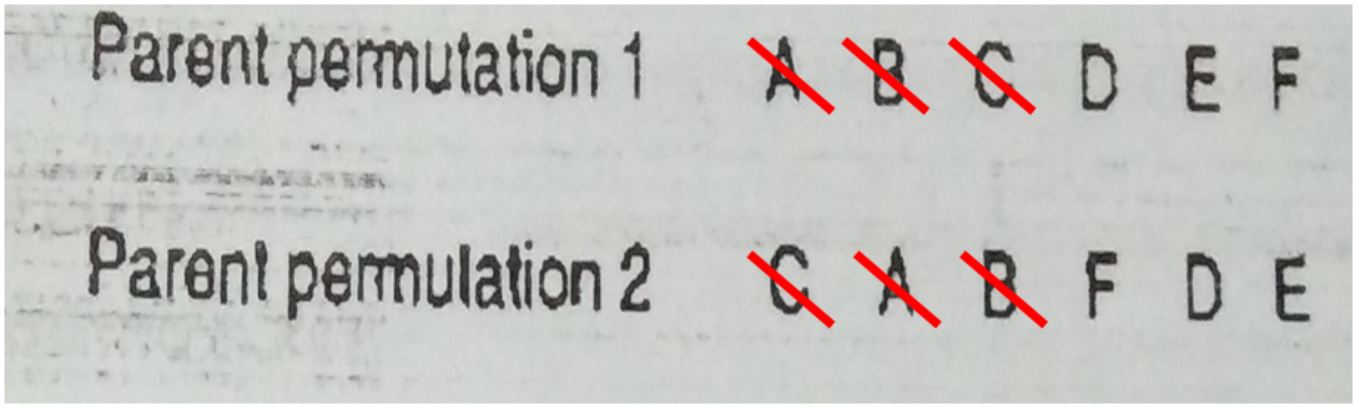


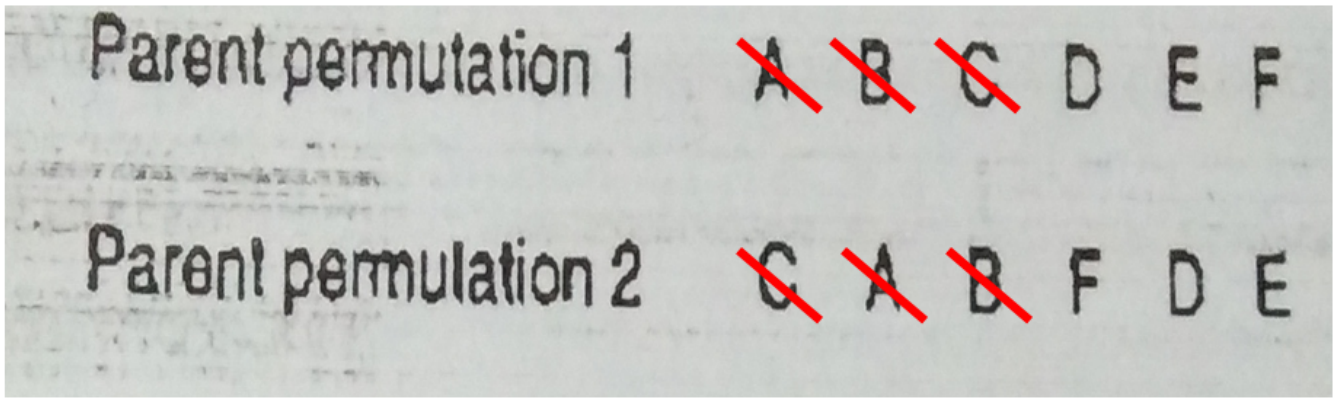
Select Parent No. (1/2)	*1*	*2*				
Offspring Permutation	*A*	*C*				



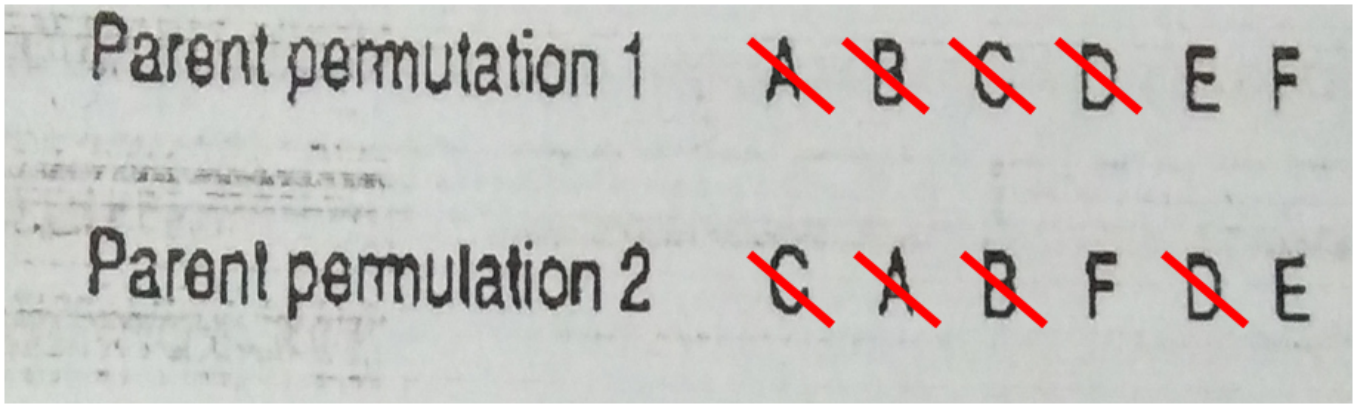


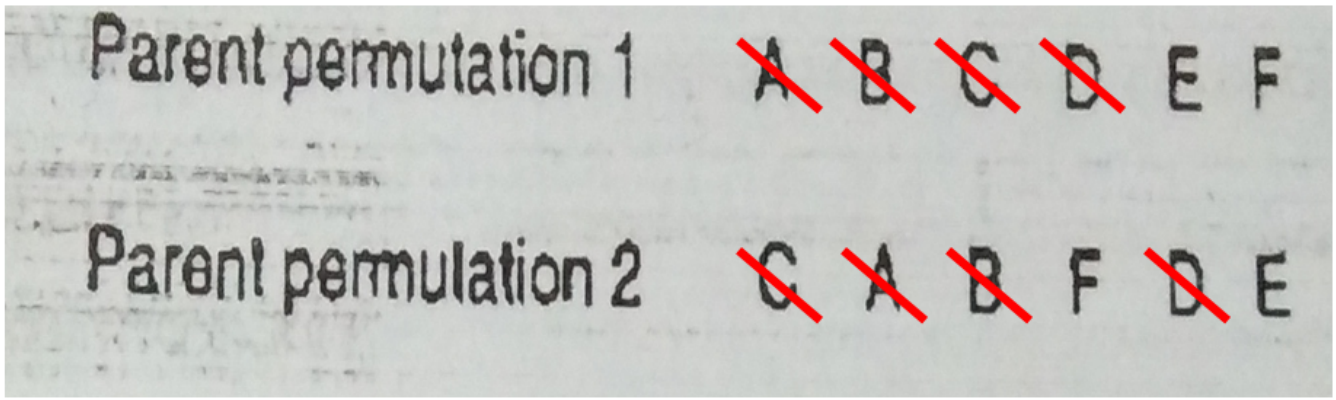
Select Parent No. (1/2)	*1*	*2*	*1*			
Offspring Permutation	*A*	*C*	*B*			



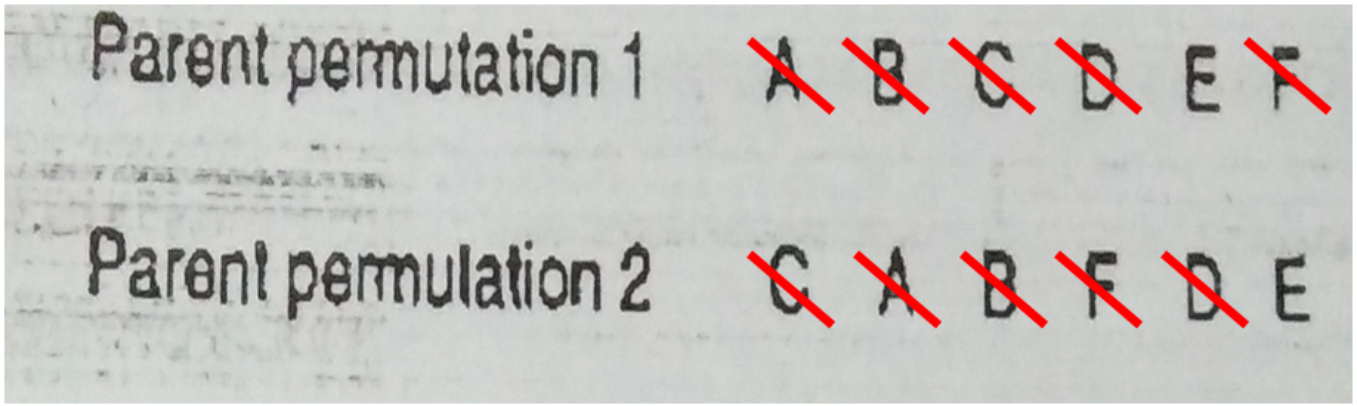


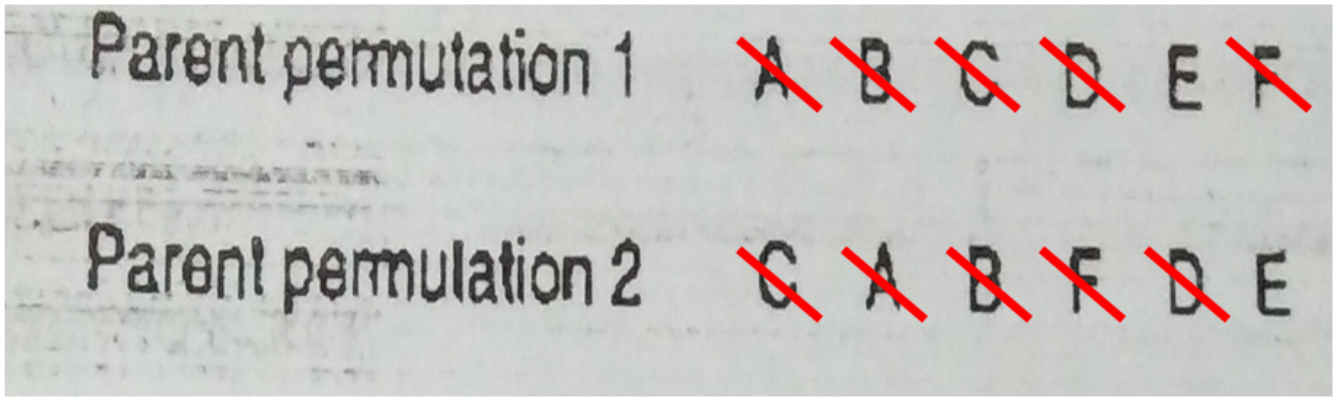
Select Parent No. (1/2)	*1*	*2*	*1*	*1*		
Offspring Permutation	*A*	*C*	*B*	*D*		



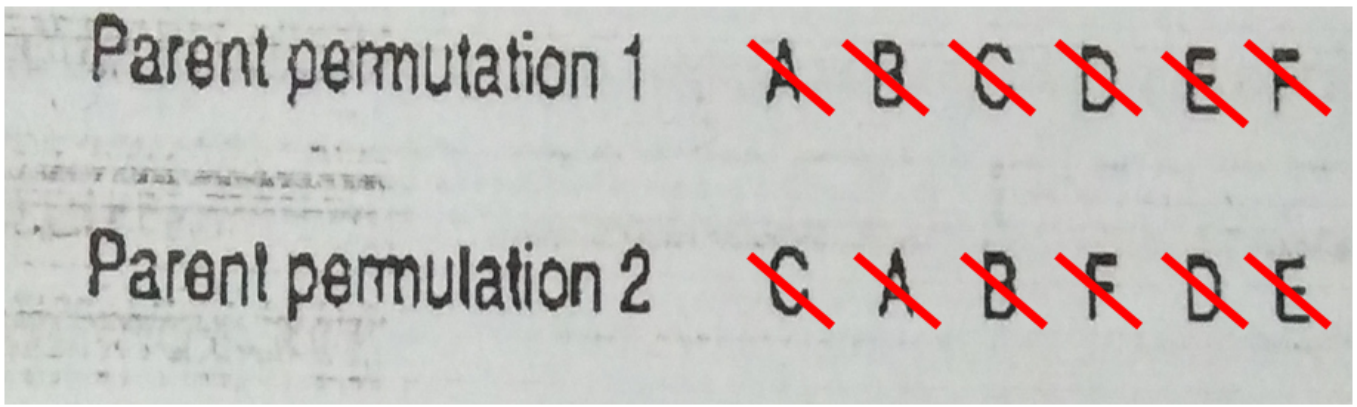


Select Parent No. (1/2)	*1*	*2*	*1*	*1*	*2*	
Offspring Permutation	*A*	*C*	*B*	*D*	*F*	





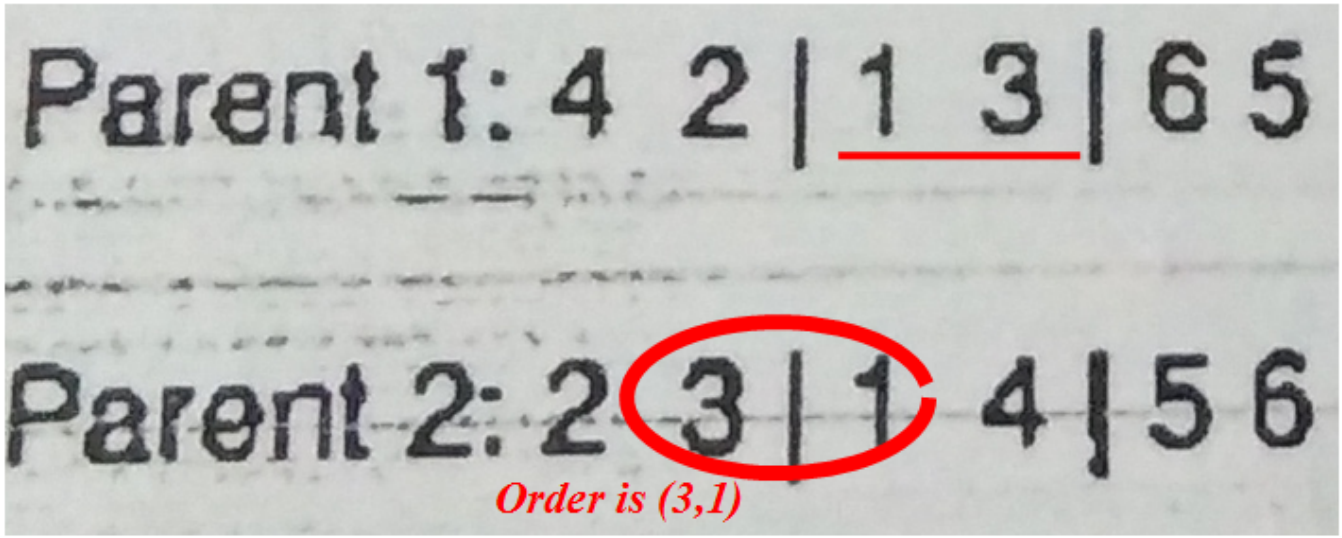
Select Parent No. (1/2)	*1*	*2*	*1*	*1*	*2*	*2*
Offspring Permutation	*A*	*C*	*B*	*D*	*F*	*E*

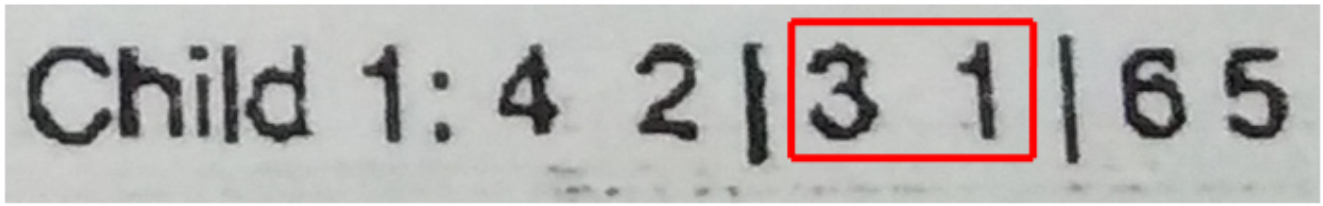


Process stop.

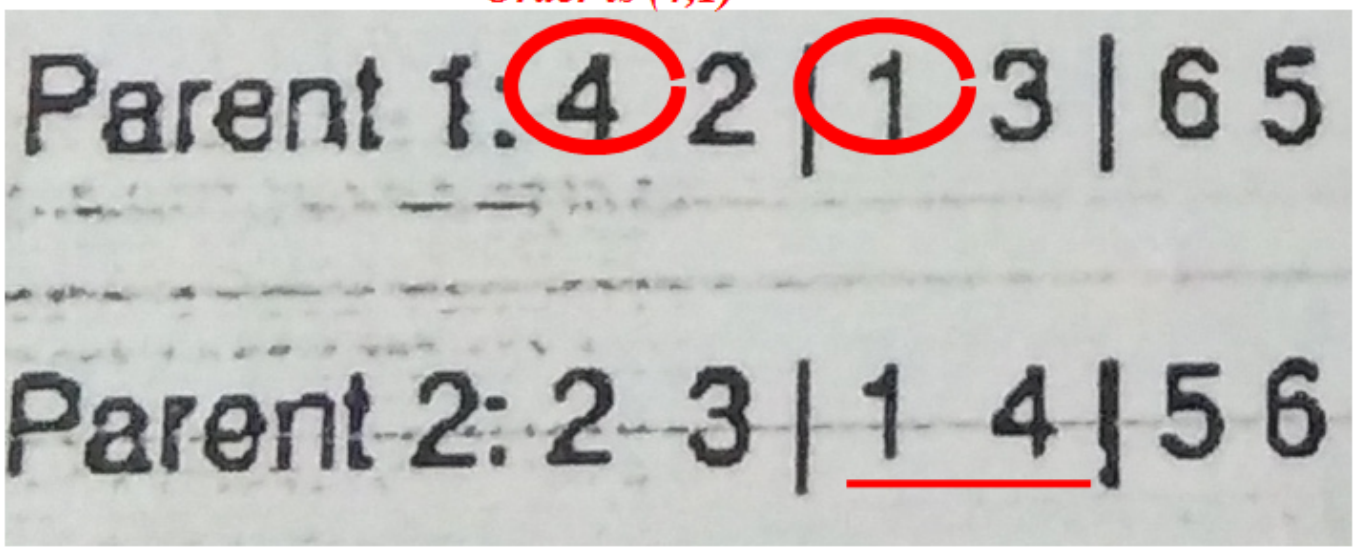
Ordered Crossover

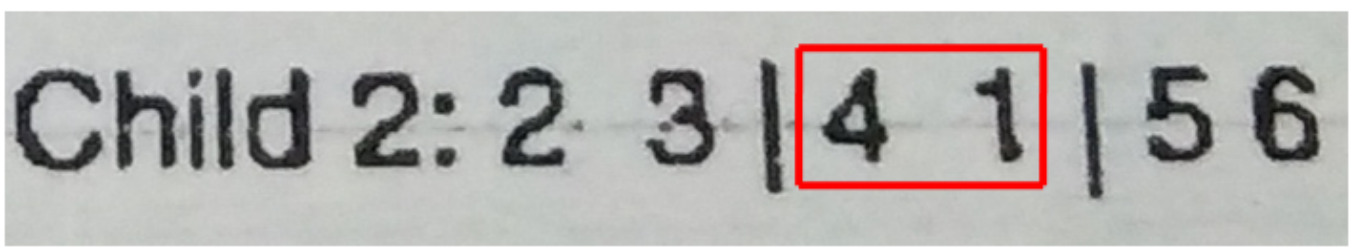
- *Ordered two point crossover* when the problem is order based.
- Given two parent chromosomes.
- Two random crossover points are selected partitioning them into a left, middle and right portions.
- *Child 1* inherits its left and right section from *parent 1* and its middle section is determined by the genes in the middle section of *parent 1* in the order in which the values appear in *parent 2*.





Order is (4,1)



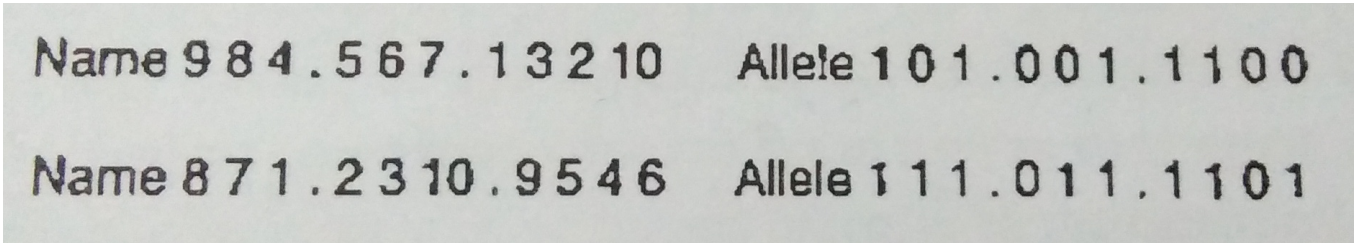


Partially Matched Crossover

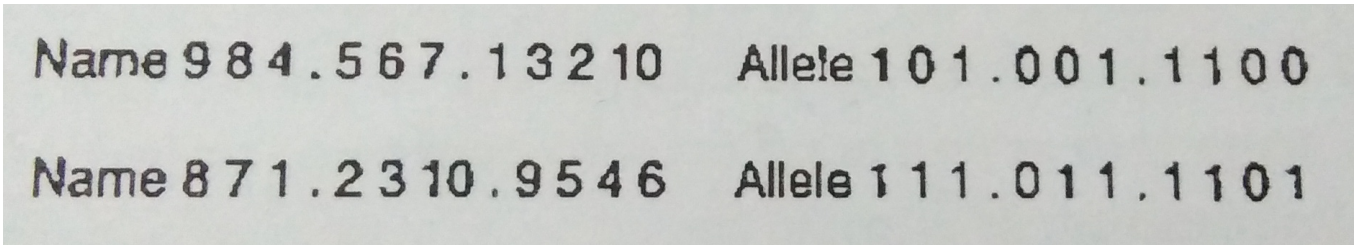
- *PMX* can be applied usefully in the *TSP*.
- *TSP* chromosomes are simply sequences of integers, where each integer represents a different city and the order represents the time at which a city is visited.
- Viewed as a crossover of permutations that guarantees that all positions are found exactly once in each offspring.
- Both offspring receive a full complement of genes, followed by the corresponding filling in of alleles from their parents.

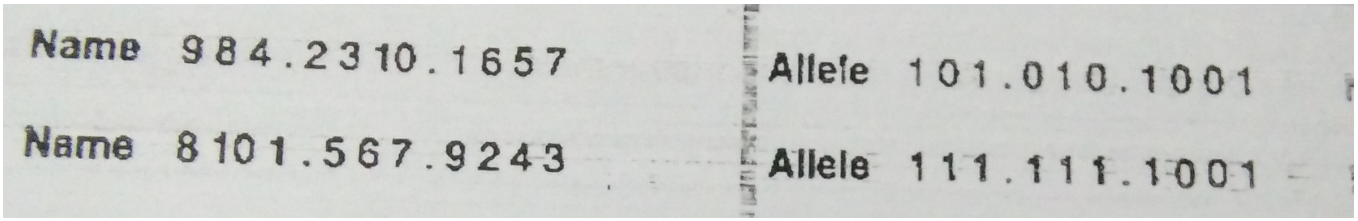
- *PMX* proceeds as follows:

1. *Two chromosomes are aligned.*
2. *Two crossing sites are selected uniformly at random along the strings, defining a matching section.*
3. *The matching section is used to effect a cross through position−by−position exchange operation.*
4. *Alleles are moved to their new positions in the offspring.*



Consider the two strings shown in figure, where the dots mark the selected cross points.
The matching section defines the position–wise exchanges that must take place in both parents to produce the offspring.
The exchanges are read from the matching section of one chromosome to that of the other.





The numbers that exchange places are,

- *5 and 2*
- *6 and 3*
- *7 and 10*

Crossover Probability

- The basic parameter in crossover technique is the *crossover probability* (P_c).
- It is a parameter to describe how often crossover will be performed.
- If there is no crossover, offspring are exact copies of parents.
- If there is crossover, offspring are made from parts of both parent's chromosome.
- If P_c is *100*%, then all offspring are made by crossover.

Mutation

- *Simple search operator*
- *Mutation* is viewed as a background operator to maintain genetic diversity in the population.

Mutation Methods

1. *Flipping*
2. *Interchanging*
3. *Reversing*
4. *Mutation Probability*

Flipping

- *Flipping* of a bit involves changing *0 to 1* and *1 to 0* based on a mutation chromosome generated.
- A parent is considered and a mutation chromosome is randomly generated.
- For *1* in mutation chromosome, the corresponding bit in parent chromosome is flipped and child chromosome is produced.

Parent	1 0 1 1 0 1 0 1
Mutation chromosome	1 0 0 0 1 0 0 1
Child	0 0 1 1 1 1 0 0

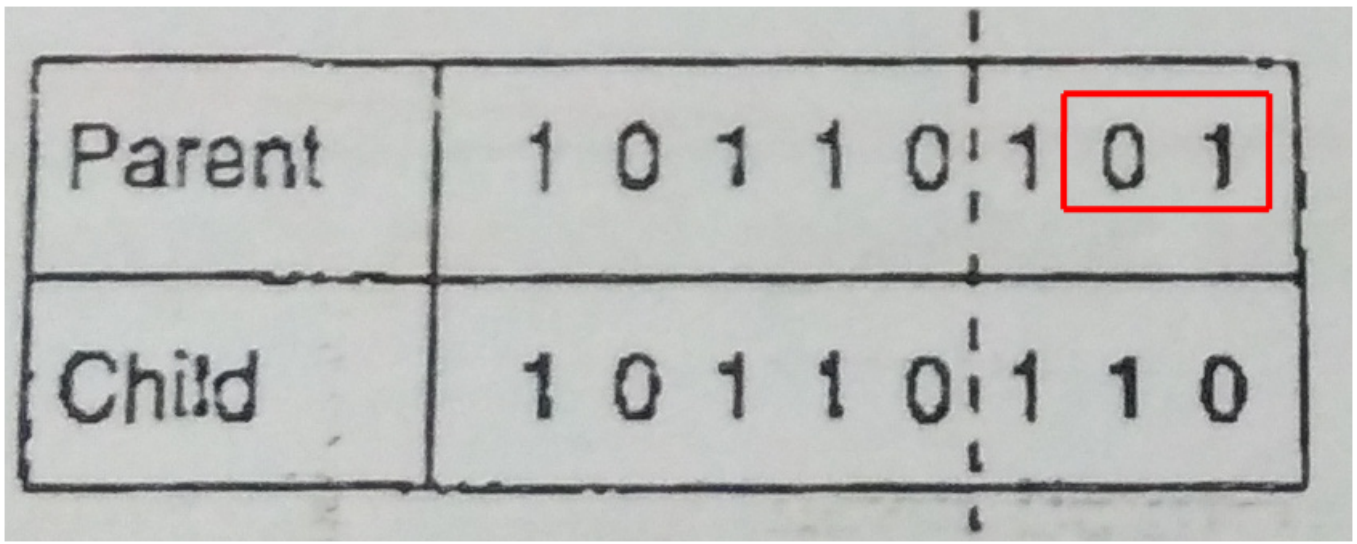
Interchanging

- Two random positions of the string are chosen and the bits corresponding to those positions are interchanged.

Parent	1 0 1 1 0 1 0 1
Child	1 1 1 1 0 0 0 1

Reversing

- A random position is chosen and the bits next to that position are reversed and child chromosome is produced.



Mutation Probability

- The basic parameter in mutation technique is the *crossover probability* (P_m).
- It is a parameter to describe how often parts of chromosome will be mutated.
- If there is no mutation, offspring are generated immediately after crossover without any change.
- If mutation is performed, one or more parts of a chromosome are changed.
- If P_m is *100*%, then whole chromosome is changed.

Stopping Condition for Genetic Algorithm Flow

- *Maximum generations*

  The GA stops when the specified number of generations has evolved.

- *Elapsed time*

  The genetic process will end when a specified time has elapsed.

  If maximum number of generation has been reached before the specified time has elapsed, then the process will end.

- *No change in fitness*

  The genetic process will end if there is no change to the population's best fitness for a specified number of generations.

  If maximum number of generation has been reached before the specified number of generation with no changes has been reached, then the process will end.

- *Stall generations*

  The GA stops if there is no improvement in the objective function for a sequence of consecutive generations of length "Stall generations".

- *Stall time limit*

  The GA stops if there is no improvement in the objective function during an interval of time in seconds equal to "Stall time limit".

Termination Techniques

1. *Best Individual*
2. *Worst Individual*
3. *Sum of Fitness*
4. *Median Fitness*

HYBRID SOFT COMPUTING TECHNIQUES

LECTURE 9

November 26, 2017

Neuro−Fuzzy Hybrid Systems

- The *Neuro−fuzzy hybridization* is widely termed as *Fuzzy Neural Network (FNN)* or *Neuro−Fuzzy System (NFS)*.
- It is a learning mechanism that utilizes the training and learning algorithms from neural networks to find parameters of a fuzzy system.
- It can also be defined as a fuzzy system that determines its parameters by processing data samples by using a learning algorithm derived from or inspired by neural network theory.
- Alternately, it is a hybrid intelligent system that fuses ANNs and fuzzy logic by combining the learning and structure of neural networks with human−like reasoning style of fuzzy systems.

- The human−like reasoning style of fuzzy systems is incorporated by NFS through the use of fuzzy sets and a linguistic model consisting of a set of *IF−THEN* fuzzy rules.
- *NFSs are universal approximators with the ability to solicit interpretable IF−THEN rules.*
- Strength of NFSs involves *interpretability versus accuracy*.

Neuro−fuzzy is divided into two areas :

1. Linguistic fuzzy modeling focused on interpretability (*Mamdani model*).
2. Precise fuzzy modeling focused on accuracy (*Sugeno model*).

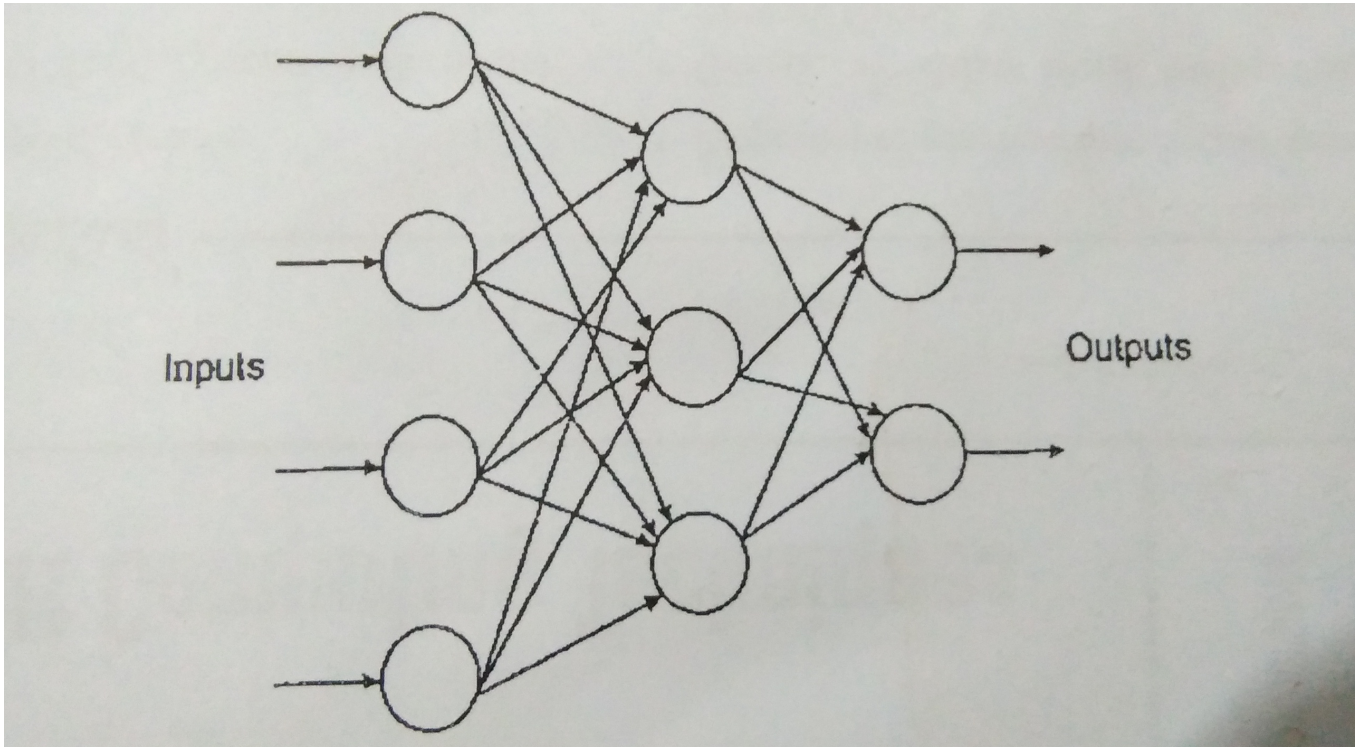
Comparison of Fuzzy Systems with Neural Networks

Neural processing	Fuzzy processing
Mathematical model not necessary	Mathematical model not necessary
Learning can be done	A priori knowledge is needed
There are several learning algorithms	Learning is not possible
Black–box behavior	Simple interpretation and implementation

Characteristics of Neuro−Fuzzy Hybrids

- A fuzzy system−based NFS is trained by means of a data−driven learning method derived from neural network theory.
- At any stage of learning process, it can be represented as a set of fuzzy rules.
- For ensuring the semantic properties of the underlying fuzzy system, the learning procedure is constrained.
- An NFS approximates an n−dimensional unknown function, partly represented by training examples.
- Fuzzy rules can be interpreted as vague prototypes of the training data.

Architecture of Neuro−Fuzzy hybrid system



An NFS is given by a three−layer feed forward neural network model. First layer corresponds to the input variables, second and third layer correspond to the fuzzy rules and output variables, respectively. The fuzzy sets are converted to connection weights.

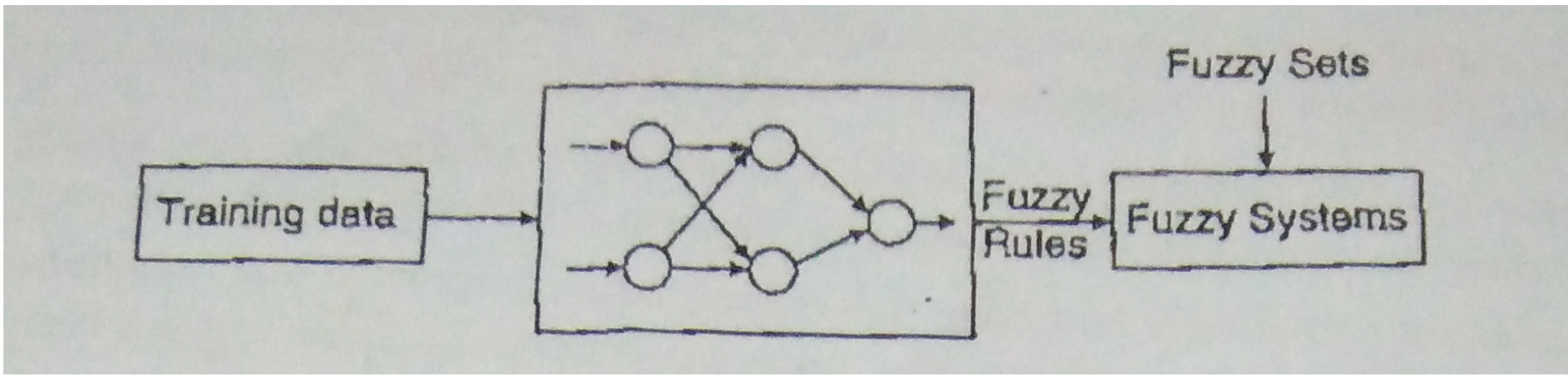
Classifications of Neuro−Fuzzy Hybrid Systems

1. *Cooperative NFSs*
2. *General neuro−fuzzy hybrid systems*

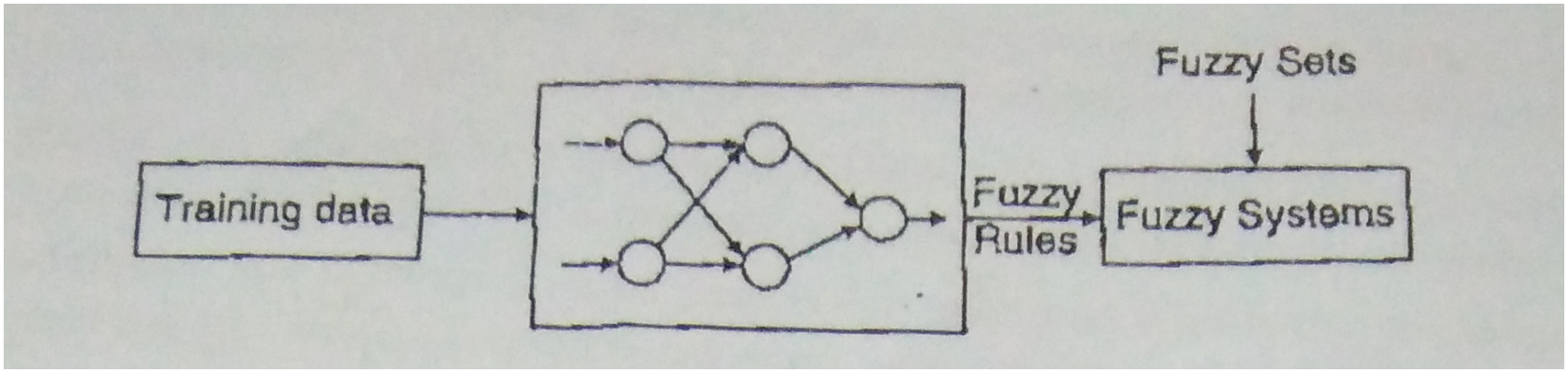
Cooperative Neural Fuzzy Systems

- Both ANN and fuzzy system work independently from each other.
- The ANN attempts to learn the parameters from the fuzzy system.
- *Four different kinds of Cooperative NFSs are there.*

Type 1

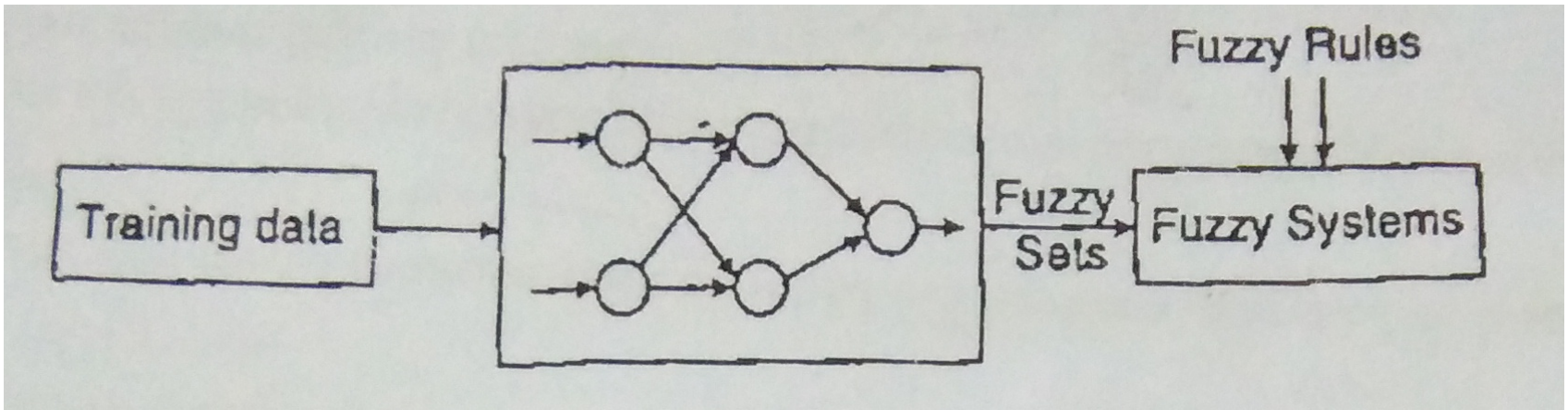


Type 1

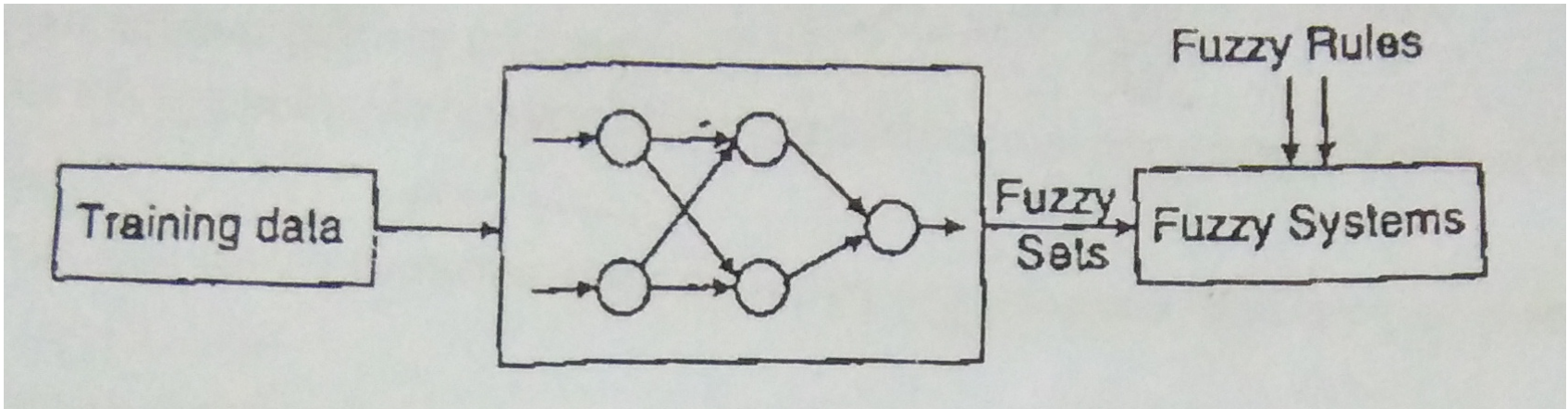


- The FNN learns fuzzy set from the given training data.
- This is done, usually, by fitting membership functions with a neural network.
- The fuzzy sets then being determined offline.
- This is followed by their utilization to form the fuzzy system by fuzzy rules that are given and not learned.

Type 2

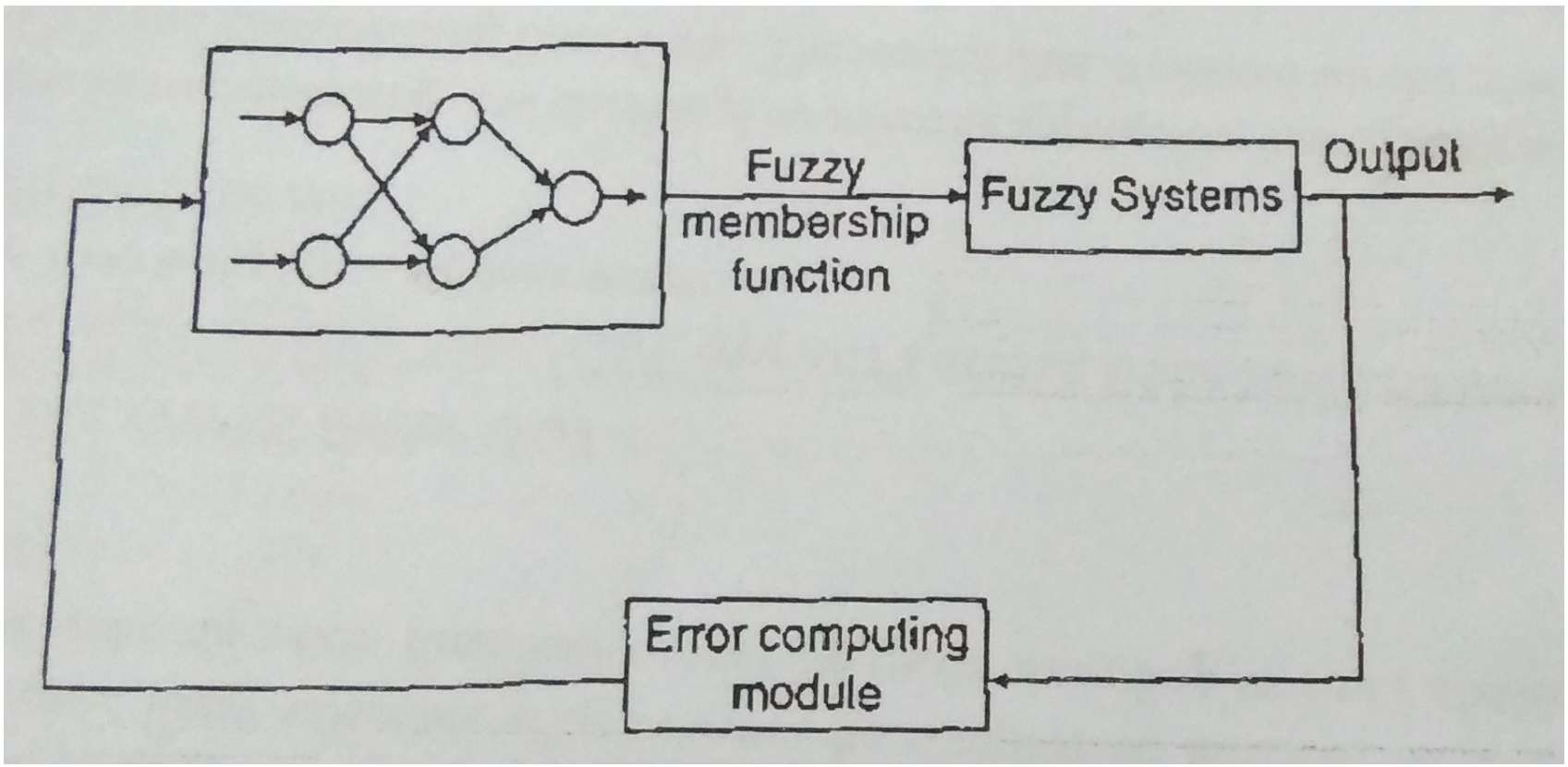


Type 2

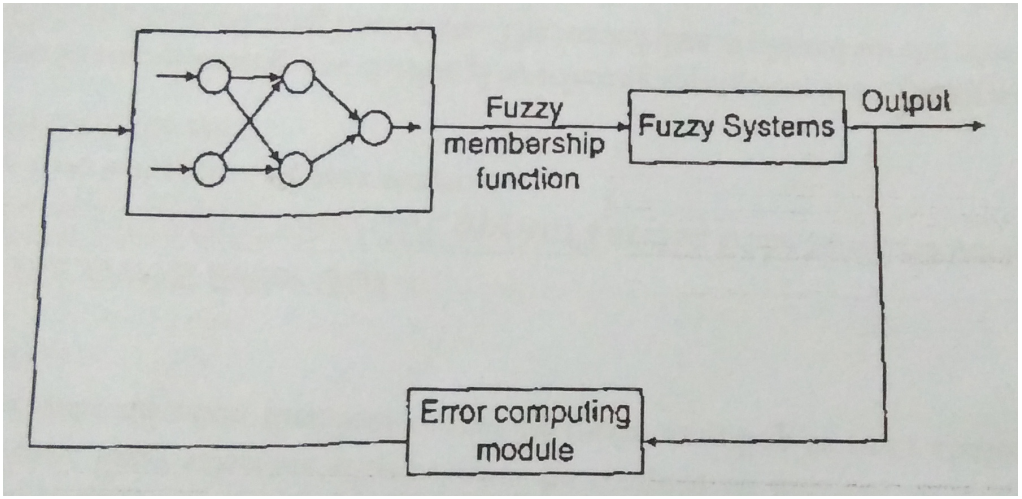


- This NFS determines, by a neural network, the fuzzy rules from the training data.
- The neural networks learn offline before the fuzzy system is initialized.

Type 3

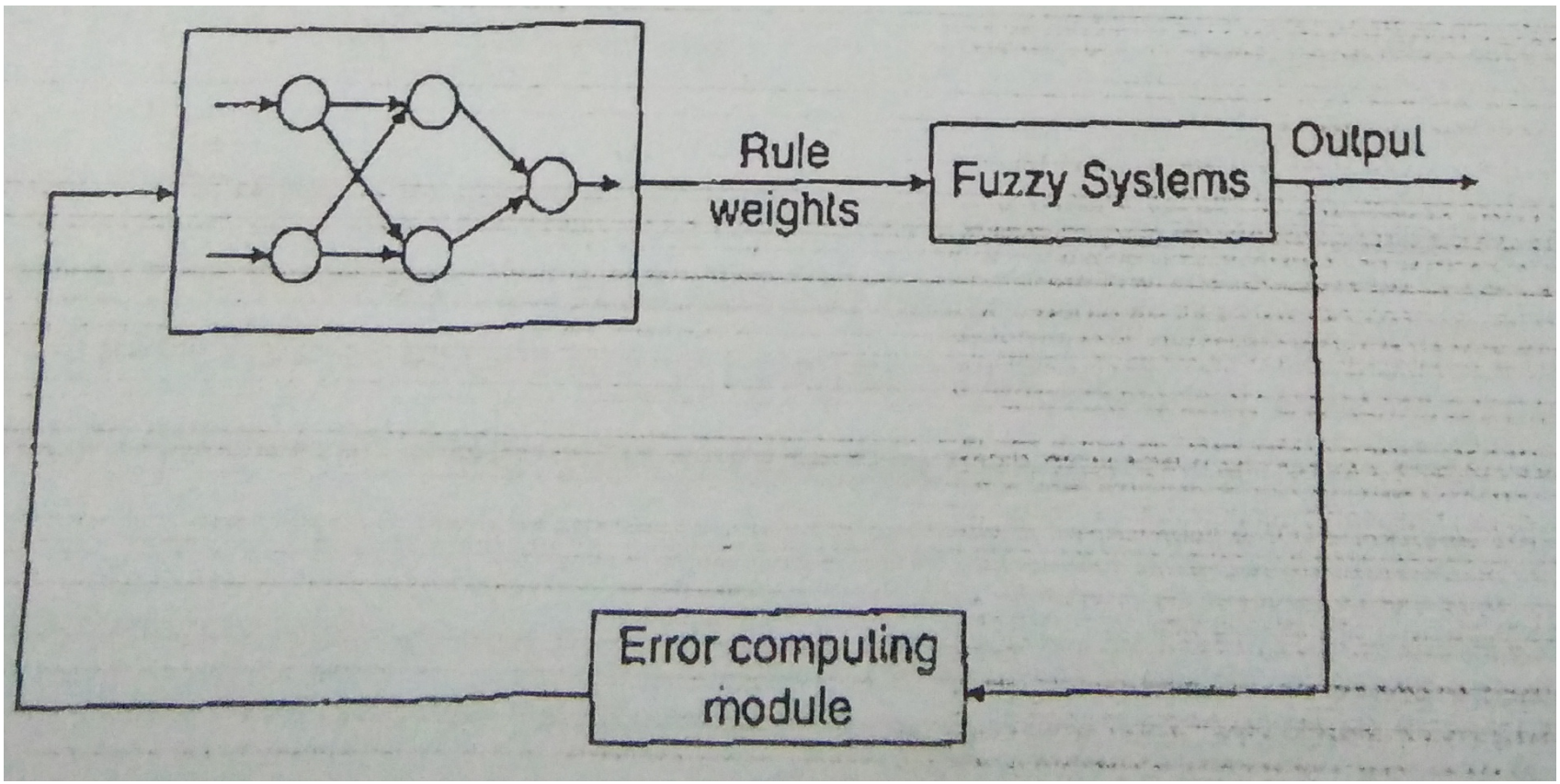


Type 3

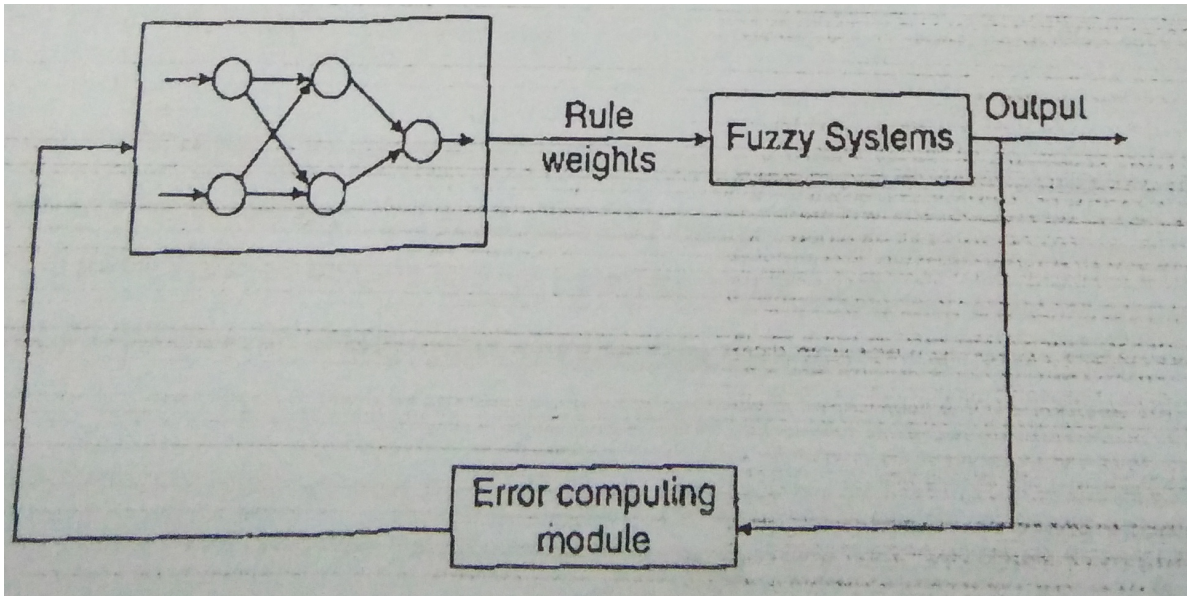


- The parameters of membership functions are learn online, while the fuzzy system is applied.
- Initially, fuzzy rules and membership functions must be defined beforehand.
- Inorder to improve and guide the learning step, the error has to be measured.

Type 4



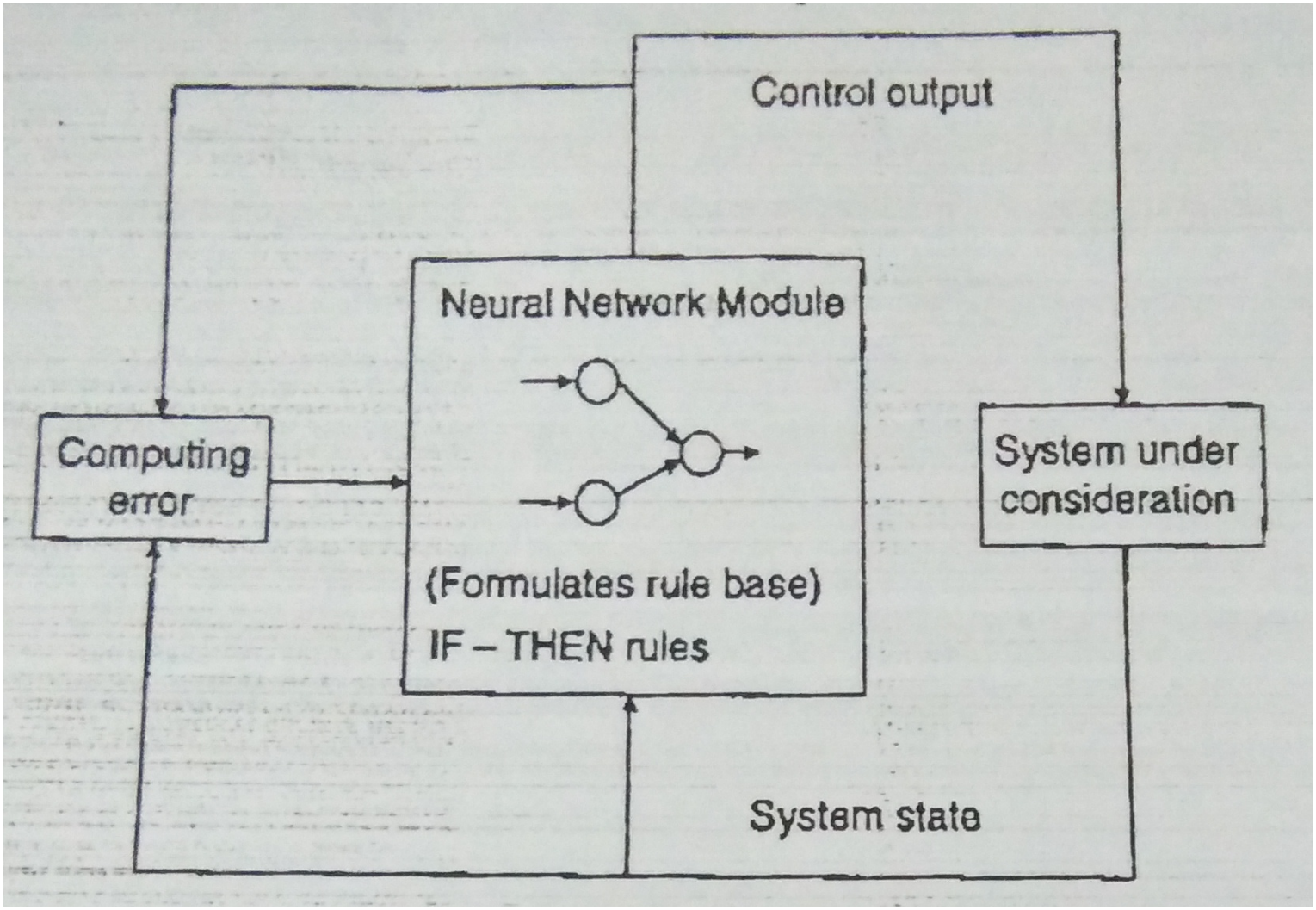
Type 4



- The model determines the rule weights for all fuzzy rules by a neural network.
- A rule is determined by its rule weights.
- They are then multiplied with the rule output.

General Neuro−Fuzzy Hybrid Systems(General NFHS)

- The *General NFHS* resemble neural networks where a fuzzy system is interpreted as a neural network of special kind.
- The architecture gives it an advantage because there is no communication between fuzzy system and neural network.



- Membership functions expressing the linguistic terms of the inference rules should be formulated for building a fuzzy controller.
- In fuzzy systems, no formal approach exists to define these functions.
- For fuzzy systems, the optimization of these functions in terms of generalizing the data is very important.
- This problem can be solved by using neural networks.

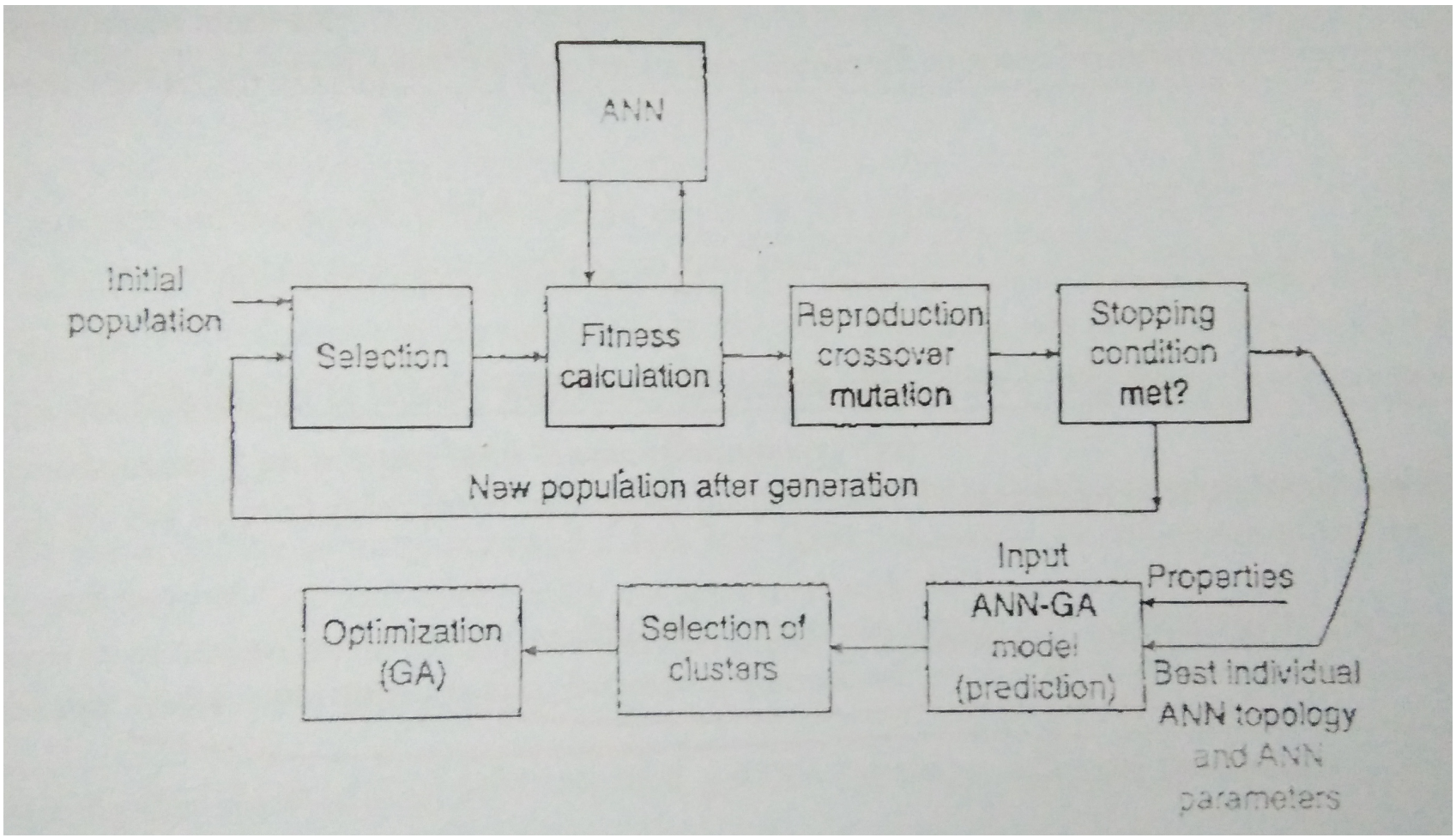
- Using learning rules, the neural network must optimize the parameters by fixing a distinct shape of the membership functions.
- Regardless of the shape of membership functions, training data should also be available.

- In neuro−fuzzy hybrid systems, the training data is grouped into several clusters.
- Each cluster is designed to represent a particular rule.
- These rules are defined by the crisp data points and are not defined linguistically.
- In this case, the neural network might be applied to train the defined clusters.
- The testing can be carried out by presenting a random testing sample to the trained neural network.
- Each and every output unit will return a degree, which extends to fit to the antecedent of rule.

Genetic Neuro−Hybrid Systems

- A *neuro−genetic hybrid or genetic neuro−hybrid system* is one in which a neural network employs a genetic algorithm to optimize its structural parameters that define its architecture.
- Neural networks solve problems by self−learning and self−organizing.
- Genetic algorithms present themselves as a potential solution for the optimization of parameters of neural networks.

Block Diagram of Genetic Neuro–Hybrids



Properties of Genetic Neuro–Hybrid Systems

- The parameters of neural networks are encoded by GAs as a string of properties of the network.
- A large population of chromosome is generated, which represent many possible parameter sets for the given neural network.
- GANN has the ability to locate neighborhood of the optimal solution quickly.

Drawbacks

- Large amount of memory required for handling and manipulation of chromosomes for a given network.
- The question of scalability of this problem as the size of the networks become large.

Genetic Algorithm Based Back−Propagation Network (*BPN*)

- *BPN* is a method of teaching multi−layer neural networks how to perform a given task.
- Here learning occurs during this training phase.
- *BPN* determines its weight based on gradient search technique.
- It may encounter a local minima problem.
- Though *GA* do not guarantee to find global optimum solution, they are good in quickly finding good acceptable solutions.

Limitations of BPN

- *BPN* do not have the ability to recognize new patterns.
- They can recognize patterns similar to those they have learn
- They must be sufficiently trained so that enough general features applicable to both seen and unseen instances can be extracted.
- There may be undesirable effects due to over training the network.

Coding

- Assume a BPN configuration $n–l–m$,where n is the number of neurons in the input layer, l is the number of neurons in the hidden layer and m is the number of output layer neurons.
- The number of weight to be determined is given by,

$$(n + m)l$$

- Each *weight*, which is a *gene* here, is a real number.

- Let d be the number of digits in weight.
- Then a string S of decimal values having string length,

$$(n+m)ld$$

  is randomly generated.
- It is a string that represents weight matrices of input−hidden and hidden−output layers in a linear form.
- Thereafter a *population of p*, which is the *population size*, chromosomes is randomly generated.

Weight Extraction

- Inorder to determine the *fitness values*, weights are extracted from each chromosome.
- Let,

$$a_1, a_2, \cdots, a_d, \cdots, a_l$$

represent a chromosome.

- Let,

$$a_{pd+1}, a_{pd+2}, \cdots, a_{(p+1)d}$$

represent p^{th} *gene* $(p \geq 0)$ in the chromosomes.

- The actual weight w_p is given by,

$$\begin{cases} -\dfrac{a_{pd+2}10^{d-2} + a_{pd+3}10^{d-3} + \cdots + a_{(p+)d}}{10^{d-2}} & if\ 0 \leq a_{pd+1} < 5 \\ +\dfrac{a_{pd+2}10^{d-2} + a_{pd+3}10^{d-3} + \cdots + a_{(p+)d}}{10^{d-2}} & if\ 5 \leq a_{pd+1} \leq 9 \end{cases}$$

Fitness Function

- A fitness has to be formulated for each and every problem to be solved.
- Consider the matrix given by,

$$\begin{bmatrix} (x_{11}, x_{21}, x_{31}, \cdots, x_{n1}) & (y_{11}, y_{21}, y_{31}, \cdots, y_{n1}) \\ (x_{12}, x_{22}, x_{32}, \cdots, x_{n2}) & (y_{12}, y_{22}, y_{32}, \cdots, y_{n2}) \\ \vdots & \vdots \\ (x_{1m}, x_{2m}, x_{3m}, \cdots, x_{nm}) & (y_{1m}, y_{2m}, y_{3m}, \cdots, y_{nm}) \end{bmatrix}$$

where X and Y are inputs and targets respectively.

- Compute initial population I_0 of size j.
- Let, $O_{10}, O_{20}, \cdots, O_{j0}$ represent j chromosomes of the initial population I_0.
- Let the weights extracted for each of the chromosomes up to j^{th} chromosome be $w_{10}, w_{20}, w_{30}, \cdots, w_{j0}$.
- For n number of inputs and m number of outputs, calculated output of the considered BPN be,

$$\begin{bmatrix} (c_{11}, c_{21}, c_{31}, \cdots, c_{n1}) \\ (c_{12}, c_{22}, c_{32}, \cdots, c_{n2}) \\ (c_{13}, c_{23}, c_{33}, \cdots, c_{n3}) \\ . \\ . \\ . \\ (c_{1m}, c_{2m}, c_{3m}, \cdots, c_{nm}) \end{bmatrix}$$

- As a result, the error here is calculated by,

$$ER_1 = (y_{11} - c_{11})^2 + (y_{21} - c_{21})^2 + (y_{31} - c_{31})^2 + \cdots + (y_{n1} - c_{n1})^2$$
$$ER_2 = (y_{12} - c_{12})^2 + (y_{22} - c_{22})^2 + (y_{32} - c_{32})^2 + \cdots + (y_{n2} - c_{n2})^2$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$ER_m = (y_{1m} - c_{1m})^2 + (y_{2m} - c_{2m})^2 + \cdots + (y_{nm} - c_{nm})^2$$

- The fitness function is derived from this *root mean square error* given by,

$$FF_n = \frac{1}{E_{rmse}}$$

- The process has to be carried out for all the total number of chromosomes.

Reproduction of Offspring

- Before the parents produce the offspring with better fitness, the mating pool has to be formulated.
- This is accomplished by *neglecting the chromosome with minimum fitness* and *replacing it with a chromosome that have maximum fitness.*
- Once the mating pool is formulated, *parent pairs* are selected randomly.
- The chromosomes of respective pairs are *combined using crossover technique* to reproduce offspring.
- The *selection operator* is used to select the best parent to participate in the reproduction process.

Convergence

- *Convergence* for *GA* is the number of generations with which the fitness value increases towards global optimum.
- It is the progression towards increasing *uniformity*.
- When about 95% of the individuals in the population share the same fitness value then the population has *converged*.

Advantages of Neuro–Genetic Hybrids

- The GA performs optimization of the NN parameters with simplicity, ease of operation, minimal requirements and global perspective.
- GA helps to find out complex structure of ANN for given input and the output data set by using its learning rule as a fitness function.
- The hybrid approach ensembles a powerful model that could significantly improve the predictability of the system under construction.

Applications

1. *Load forecasting*
2. *Stock forecasting*
3. *Cost optimization in textile industries*
4. *Medical diagnosis*
5. *Face recognition*
6. *Multi–processor scheduling*

Genetic Fuzzy Hybrid and Fuzzy Genetic Hybrid Systems

- This integration can be performed in two ways:
  1. By the use of fuzzy logic based techniques for improving generic algorithm behavior and modeling GA components. This is called *fuzzy genetic algorithms (FGA)*.
  2. By the application of genetic algorithms in various optimization and search problems involving fuzzy systems.

- An *FGA* is considered as a GA that uses techniques or tools based on fuzzy logic to improve the GA behavior modeling.
- It may also be defined as an ordering sequence of instructions in which some of the instructions or algorithm components may be designed with tools based on fuzzy logic.
- The GAs are utilized for solving different fuzzy optimization problems.

Genetic Fuzzy Rule Based Systems (*GFRBSs*)

- For modeling complex systems in which the classical tools are unsuccessful due to they are complex or imprecise, thus an important tool in the form of fuzzy rule based systems has been identified.
- For mechanizing the definition of the knowledge base of a fuzzy controller GAs have proven to be a powerful tool, since adaptive control, learning, and self-organization may be considered in a lot of cases as optimization or search processes.
- In general these approaches can be termed as *Genetic Fuzzy Systems (GFSs)*.

Block Diagram of Genetic Fuzzy System

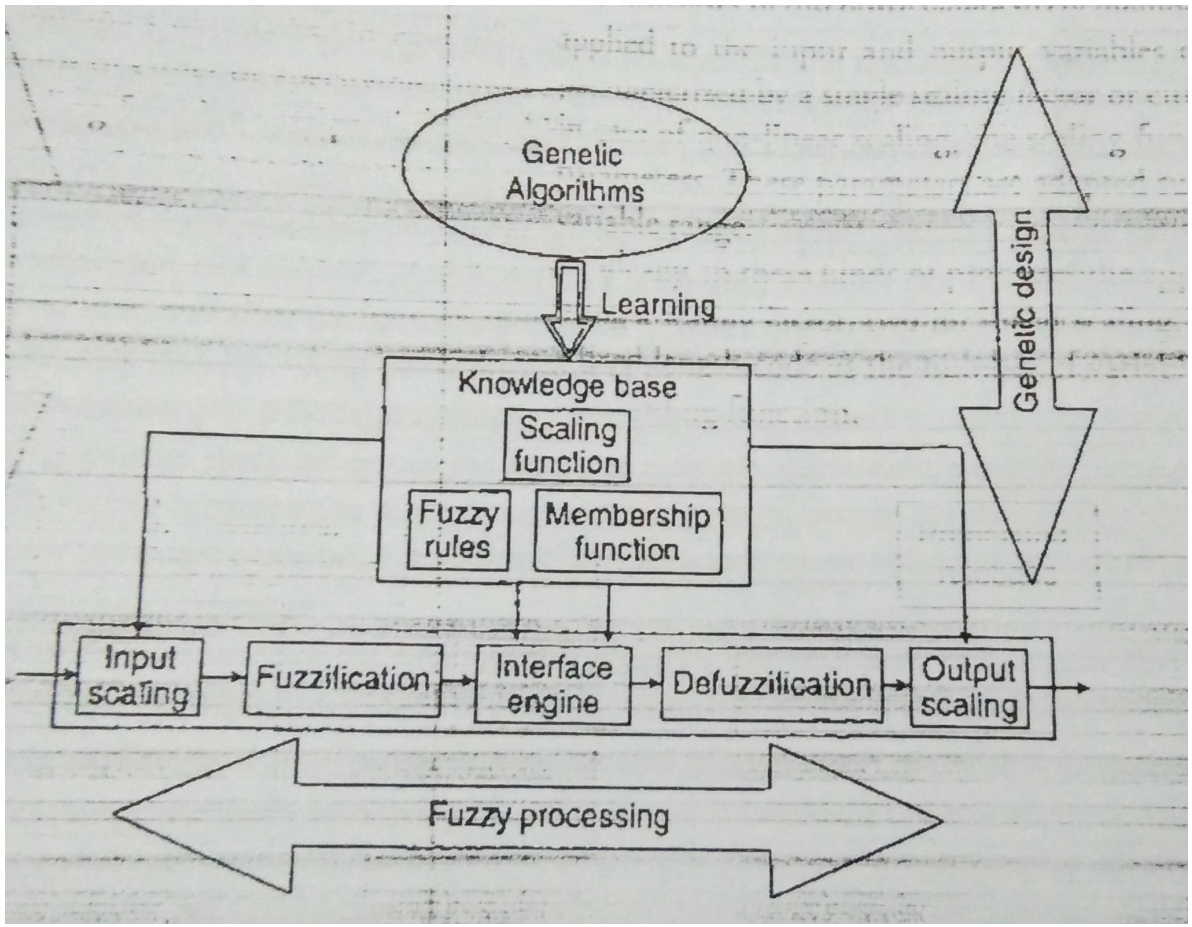


Figure shows a system where genetic design and fuzzy processing are the two fundamental constituents.

- Inside *GFRBSs*, it is possible to distinguish between either parameter optimization or rule generation processes, that is, adaptation and learning.
- The main objectives of optimization in *fuzzy rule based system* are as follows:

1. The task of finding an appropriate *knowledge base (KB)* for a particular problem. This is equivalent to parameterizing the *fuzzy KB (rules and membership functions)*.

2. To find those parameter values that are optimal with respect to the design criteria.

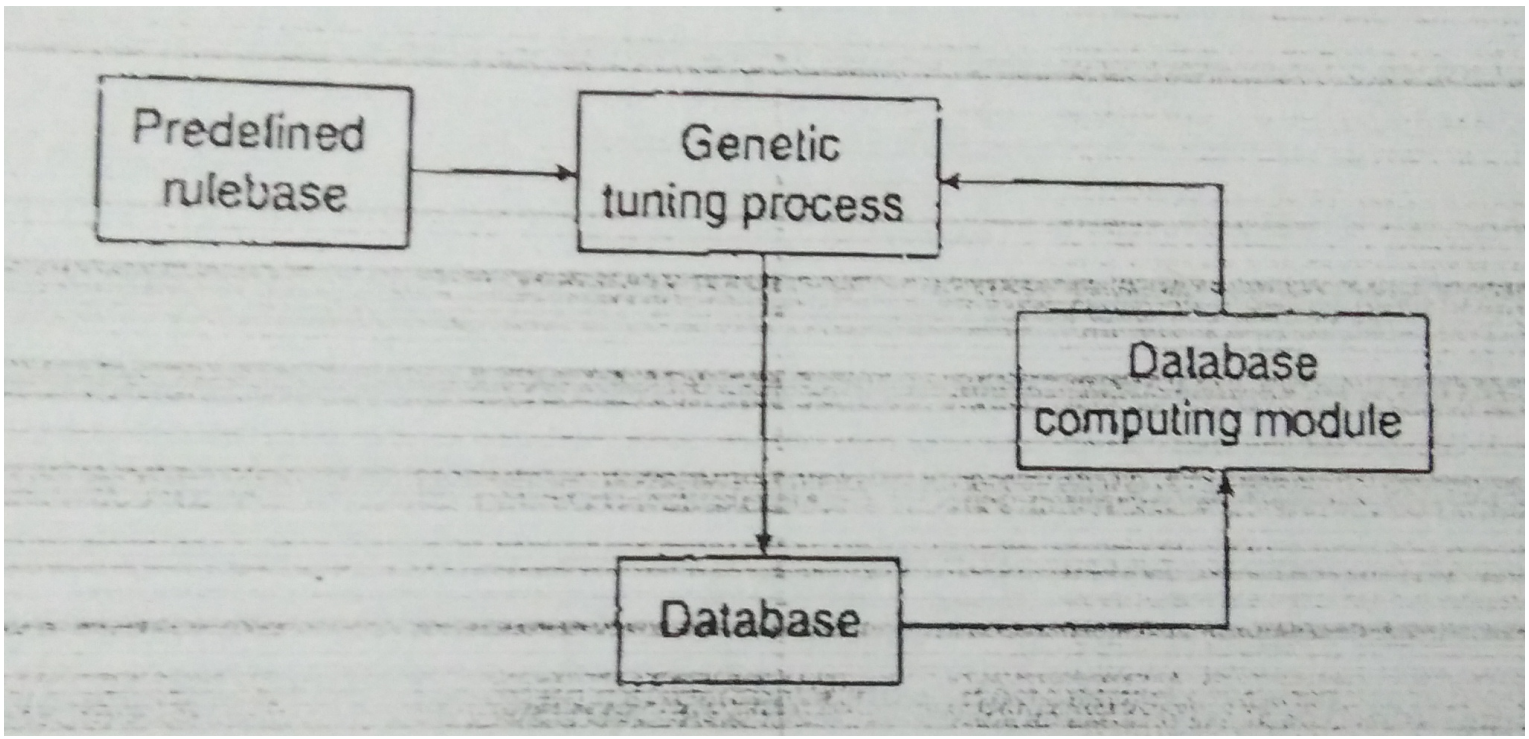
Tuning versus Learning problems

Tuning	Learning
It is concerned with optimization of an existing FRBS.	It constitutes an automated design method for fuzzy rule sets that start from scratch
Tuning processes assume a predefined RB and have the objective to find a set of optimal parameters for the membership and/or DB parameters.	Learning processes perform a more elaborated search in the space of possible RBs or whole KB and do not depend on a predefined set of rules.

Genetic Tuning Process

- The task of tuning the scaling functions and fuzzy membership functions is important in FRBS design.
- Adoption of parameterized scaling functions and membership functions by GA is based on the fitness function that specifies the design criteria quantitatively.
- The responsibility of finding a set of optimal parameters for the membership and/or the scaling functions rests with the tuning processes which assume a predefined rule base.
- The tuning process can he performed *a priori* also.
- This can be done if a subsequent process derives the RB once the DB has been obtained, *ie*, *a priori* genetic DB learning.

Process of tuning the DB



Tuning Scaling Functions

- The universes of discourse where fuzzy membership functions are defined are normalized by scaling functions applied to the input and output variables of FRBSs.
- In case of linear scaling, the scaling functions are parameterized by a single scaling factor or either by specifying a lower and upper bound.
- In case of non−linear scaling, the scaling functions are then parameterized by one or several contradiction or dilation parameters.
- These parameters are adapted such that the scaled universe of discourse matches the underlying variable range.

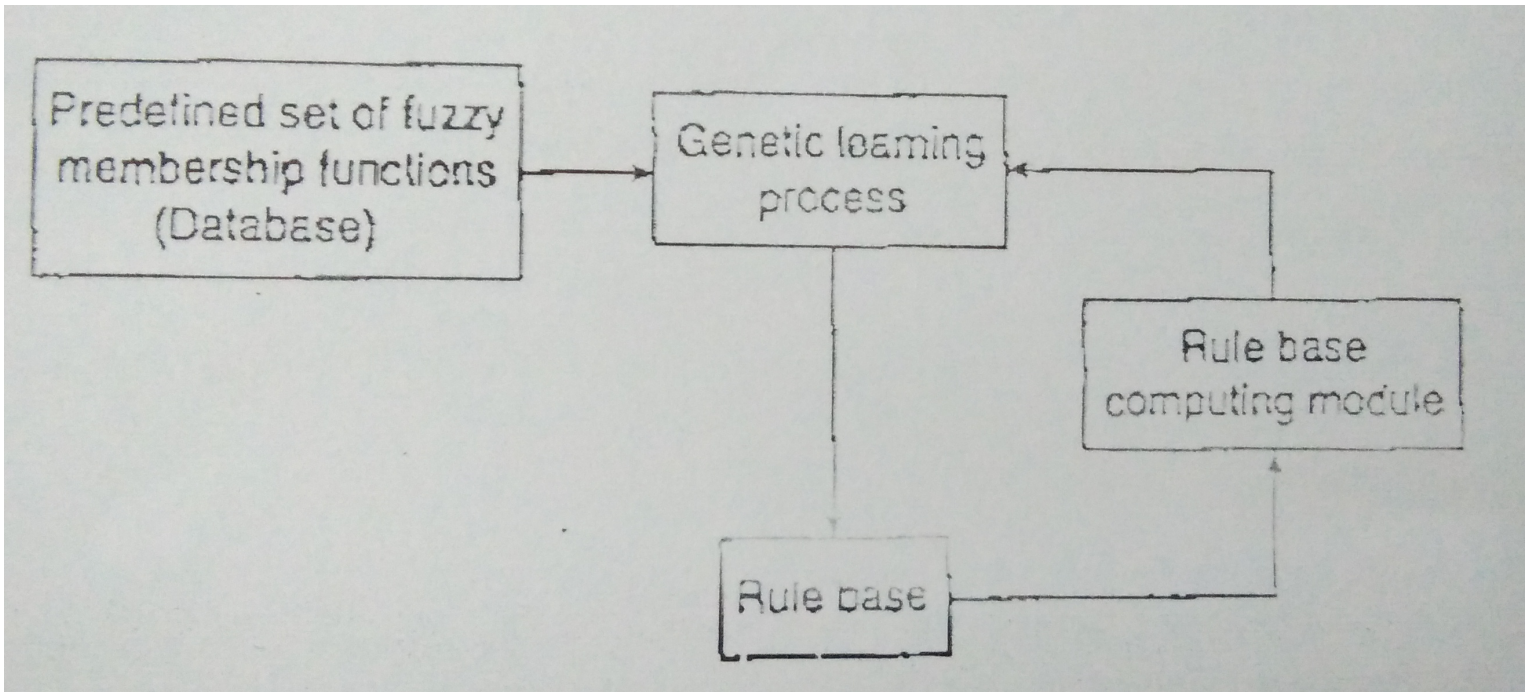
- These kinds of processes the approach is to adapt one to four parameters per variable:
  1. *using a scaling factor*
  2. *for linear scaling*
  3. *for non−linear scaling*
  4. *for non−linear scaling*
- This approach leads to a fixed length code as the number of variables is predefined as the number of parameters required to code each scaling function.

Tuning Membership Functions

- During the tuning of membership functions , an individual represents the entire DB.
- Its chromosome encodes the parameterized membership functions associated to the linguistic terms in every fuzzy partition considered by the fuzzy rule based system.
- The number of parameters per membership function can vary from one to four and each parameter can be either binary or real coded.

- For FRBSs of the *descriptive(using linguistic variables)* or *approximate(using fuzzy variables)* type, the structure of chromosome is different.
- In linguistic model, the entire fuzzy partitions are encoded into the chromosome .
- Here, the number of parameters to code is reduced to the ones defining the core regions of the fuzzy sets.

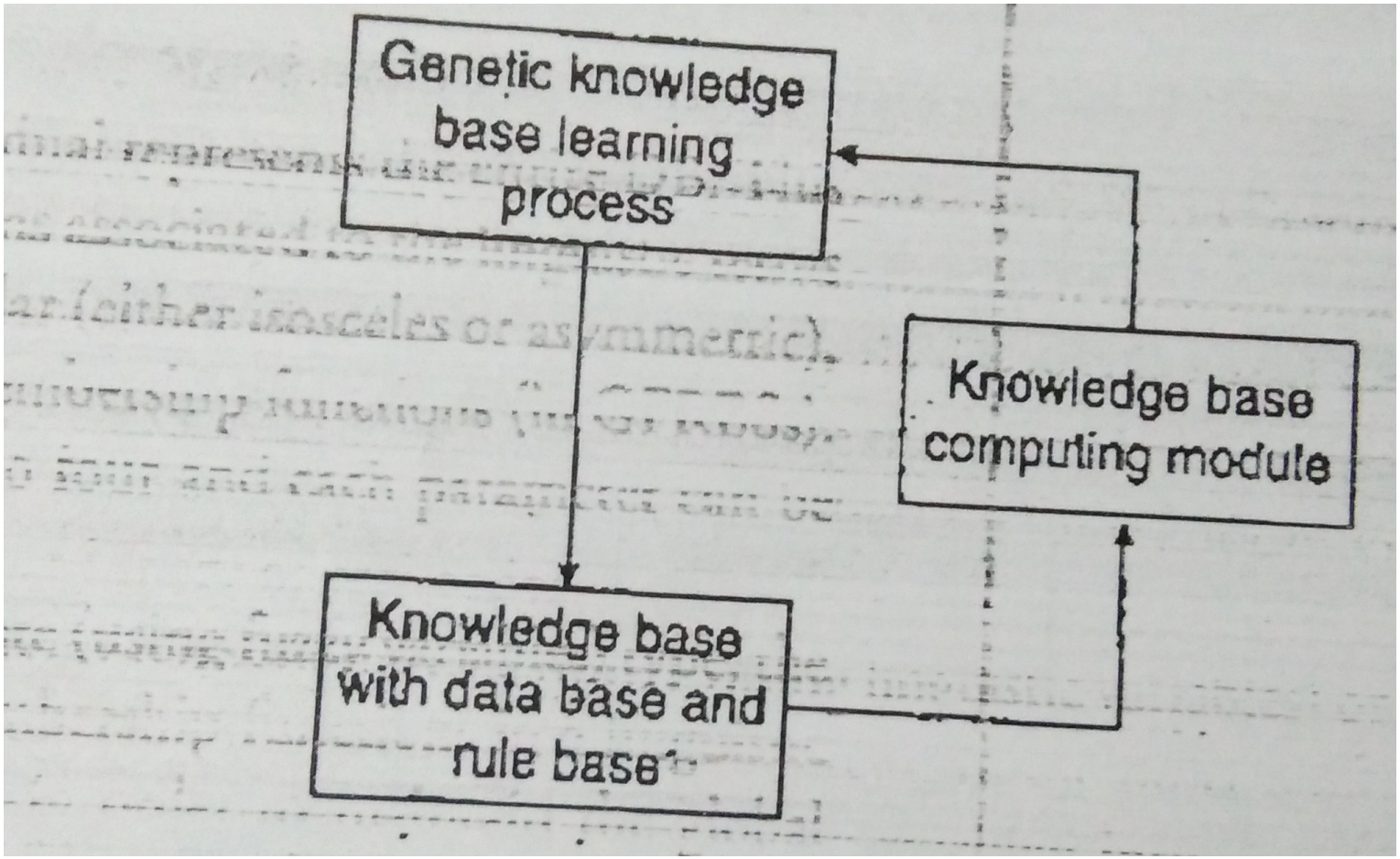
Genetic Learning of Rule Bases



- Genetic learning of rule bases assumes a predefined set of fuzzy membership functions in the DB to which the rules refer, by means of linguistic labels.
- As in the approximate approach adapting rules, it only applies to descriptive FRBSs, which is equivalent to modifying the membership functions.
- When considering a rule based system and focusing on learning rules, there are three main approaches:
  1. *Pittsburgh approach*
  2. *Michigan approach*
  3. *Iterative rule learning approach*

- The *Pittsburgh approach* is characterized by representing an entire rule set as a generic code (*chromosome*), maintaining a population of candidate rule sets and using selection and generic operators to produce new generations of rule sets.
- The *Michigan approach* considers a different model where the members of the population are individual rules and a rule set is represented by the entire population.
- In the *Iterative rule learning approach*, the iterative one, chromosomes code individual rules, and a new rule is adapted and added to the rule set, in an iterative fashion, in every run of the genetic algorithm.

Genetic Learning of Knowledge Base



- Genetic learning of a KB includes different genetic representations such as variable length chromosomes, multi−chromosome genomes and chromosomes encoding single rules instead of a whole KB as it deals with the heterogeneous search spaces.
- As the complexity of search space increases, computational cost of the generic search also grows.
- To maintain a GFRBS that encodes individual rules rather than entire KB.
- The three learning approaches as used in case of rule base can also be considered here:
  1. *Pittsburgh approach*
  2. *Michigan approach*
  3. *Iterative rule learning approach*

Advantages of Genetic Fuzzy Hybrids

- GAs allow us to represent different kinds of structures, such as weights, features together with rule parameters allowing us to code multiple models of knowledge representation.
- This provides a wide variety of approaches where it is necessary to design specific generic components for evolving a specific representation.
- Genetic algorithm efficiently optimizes the rules, membership functions, DB and KB of fuzzy systems.
- The methodology adopted is simple and the fittest individual is identified during the process.

END