**Aim**: Design a distributed application using RPC for remote computation where client submits an integer value to the server and server calculates factorial and returns the result to the client program.

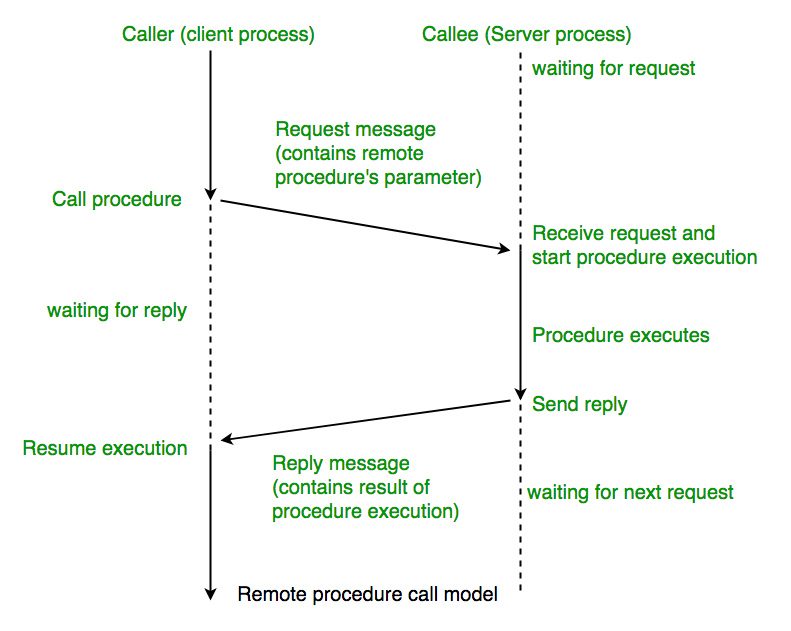
* **Outcome:** At end of this experiment, student will be able understand remote Procedure call
* **Hardware Requirement:** Computer System, Linux(Ubantu)
* **Software Requirement:** Java(JDK) & PyCharm IDE

**Theory:**

Remote Procedure Call (RPC) is a powerful technique for constructing distributed, client-server based applications. It is based on extending the conventional local procedure calling so that the called procedure need not exist in the same address space as the calling

procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them.

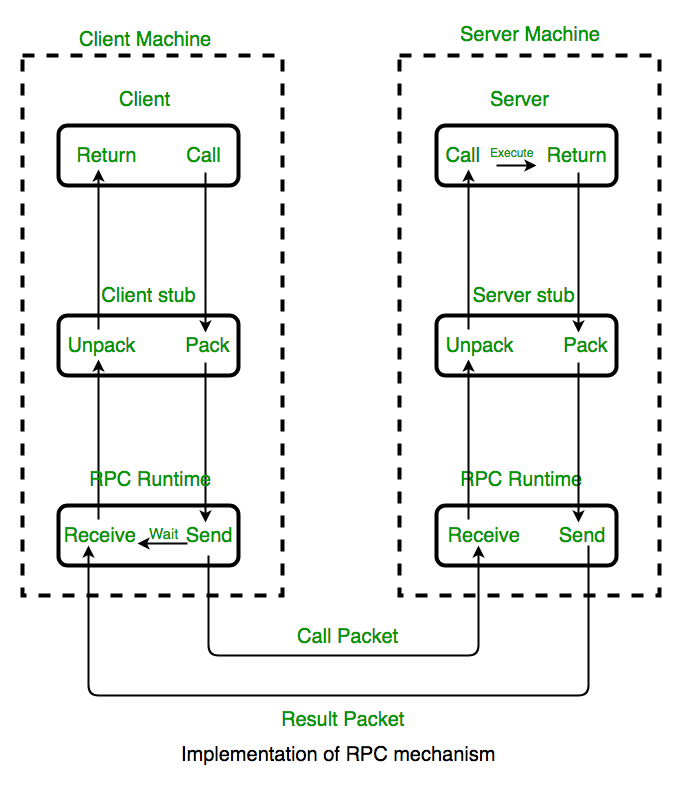
When making a Remote Procedure Call:



1. The calling environment is suspended, procedure parameters are transferred across the network to the environment where the procedure is to execute, and the procedure is executed there.

2. When the procedure finishes and produces its results, its results are transferred back to the calling environment, where execution resumes as if returning from a regular procedure call.

NOTE: RPC is especially well suited for client-server (e.g. query-response) interaction in which the flow of control alternates between the caller and callee. Conceptually, the client and server do not both execute at the same time. Instead, the thread of execution jumps from the caller to the callee and then back again. \



**Working Of RPC:**

**The following steps take place during a RPC :**

A client invokes a client stub procedure, passing parameters in the usual way. The client stub resides within the client’s own address space.

The client stub marshalls(pack) the parameters into a message. Marshalling includes converting the representation of the parameters into a standard format, and copying each parameter into the message.

The client stub passes the message to the transport layer, which sends it to the remote server machine.

On the server, the transport layer passes the message to a server stub, which demarshalls(unpack) the parameters and calls the desired server routine using the regular procedure call mechanism.

When the server procedure completes, it returns to the server stub (e.g., via a normal procedure call return), which marshalls the return values into a message. The server stub then hands the message to the transport layer.

The transport layer sends the result message back to the client transport layer, which hands the message back to the client stub.

The client stub demarshalls the return parameters and execution returns to the caller.

**Key Considerations for Designing and Implementing RPC Systems are:**

**Security:** Since RPC involves communication over the network, security is a major concern. Measures such as authentication, encryption, and authorization must be implemented to prevent unauthorized access and protect sensitive data.

**Scalability:** As the number of clients and servers increases, the performance of the RPC system must not degrade. Load balancing techniques and efficient resource utilization are important for scalability.

**Fault tolerance:** The RPC system should be resilient to network failures, server crashes, and other unexpected events. Measures such as redundancy, failover, and graceful degradation can help ensure fault tolerance.

**Standardization:** There are several RPC frameworks and protocols available, and it is important to choose a standardized and widely accepted one to ensure interoperability and compatibility across different platforms and programming languages.

**Performance tuning:** Fine-tuning the RPC system for optimal performance is important. This may involve optimizing the network protocol, minimizing the data transferred over the network, and reducing the latency and overhead associated with RPC calls.

**RPC ISSUES :**Issues that must be addressed:

1. RPC Runtime:   
RPC run-time system is a library of routines and a set of services that handle the network communications that underlie the RPC mechanism. In the course of an RPC call, client-side and server-side run-time systems’ code handle binding, establish communications over an appropriate protocol, pass call data between the client and server, and handle communications errors.

2. Stub:   
The function of the stub is to provide transparency to the programmer-written application code.

On the client side, the stub handles the interface between the client’s local procedure call and the run-time system, marshalling and unmarshalling data, invoking the RPC run-time protocol, and if requested, carrying out some of the binding steps.

On the server side, the stub provides a similar interface between the run-time system and the local manager procedures that are executed by the server.

3. Binding: How does the client know who to call, and where the service resides?   
The most flexible solution is to use dynamic binding and find the server at run time when the RPC is first made. The first time the client stub is invoked, it contacts a name server to determine the transport address at which the server resides.

**Binding consists of two parts:**

Naming:

Locating:

A Server having a service to offer exports an interface for it. Exporting an interface registers it with the system so that clients can use it.

A Client must import an (exported) interface before communication can begin.

4. The call semantics associated with RPC :  
It is mainly classified into following choices-

Retry request message –  
Whether to retry sending a request message when a server has failed or the receiver didn’t receive the message.

Duplicate filtering –  
Remove the duplicate server requests.

**Retransmission of results –**  
To resend lost messages without re-executing the operations at the server side.

**ADVANTAGES :**

RPC provides ABSTRACTION i.e message-passing nature of network communication is hidden from the user.

RPC often omits many of the protocol layers to improve performance. Even a small performance improvement is important because a program may invoke RPCs often.

RPC enables the usage of the applications in the distributed environment, not only in the local environment.

With RPC code re-writing / re-developing effort is minimized.

Process-oriented and thread oriented models supported by RPC.

**Conclusion: -** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-------------------------------------------------------------------------------------------------------------------------

**Questions:**

Q1. Explain RPC w.r.t Distributed?

Q2. Explain RPC Implementation mechanism?

Q3. Define Marshalls & DeMarshalls interms of RPC?

Q4. What Are The Issues With RPC?

**Assignment No. 2 (Part I)**

**Aim**: Design a distributed application using RMI for remote computation where client submits two strings to the server and server returns the concatenation of the given strings.

* **Outcome:** At end of this experiment, student will be able understand remote Procedure call
* **Hardware Requirement:** Computer System, Linux(Ubantu)
* **Software Requirement:** Java(JDK) & PyCharm IDE

**Theory:**

**RMI (Remote Method Invocation):**

The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects stub and skeleton.

### Understanding stub and skeleton

RMI uses stub and skeleton object for communication with the remote object.

A **remote object** is an object whose method can be invoked from another JVM. Let's understand the stub and skeleton objects:

### **stub**

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

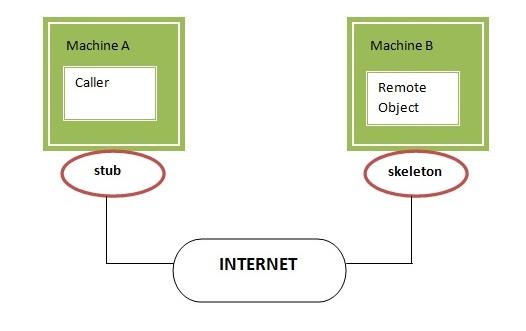
### **skeleton**

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

It reads the parameter for the remote method

1. It invokes the method on the actual remote object, and
2. It writes and transmits (marshals) the result to the caller.

In the Java 2 SDK, an stub protocol was introduced that eliminates the need for skeletons.



### Understanding requirements for the distributed applications

If any application performs these tasks, it can be distributed application.

.

1. The application need to locate the remote method
2. It need to provide the communication with the remote objects, and
3. The application need to load the class definitions for the objects.

The RMI application have all these features, so it is called the distributed application.

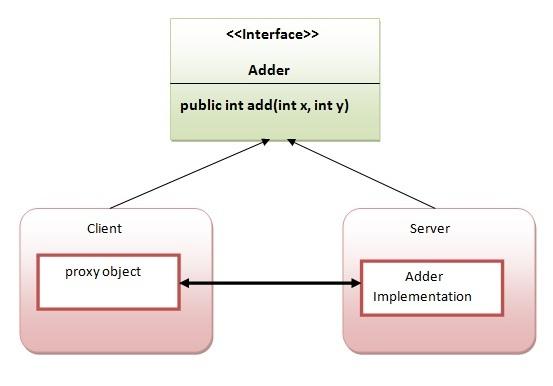
### **Java RMI Example**

The is given the 6 steps to write the RMI program.

1. Create the remote interface
2. Provide the implementation of the remote interface
3. Compile the implementation class and create the stub and skeleton objects using the rmic tool
4. Start the registry service by rmiregistry tool
5. Create and start the remote application
6. Create and start the client application

### **RMI Example**

In this example, we have followed all the 6 steps to create and run the rmi application. The client application need only two files, remote interface and client application. In the rmi application, both client and server interacts with the remote interface. The client application invokes methods on the proxy object, RMI sends the request to the remote JVM. The return value is sent back to the proxy object and then to the client application.



**Conclusion: -** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-------------------------------------------------------------------------------------------------------------------------

**Questions:**

Q1. What RMI ?How its working?

Q2. Describe JAVA RMI as an Example ?

Q3. What is role of stub & skeleton in RMI?

Q4. Differnce Between RPC & RMI?

**Assignment No. 3 (Group A)**

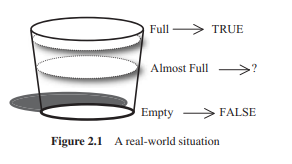
**Aim**: Design a distributed application using RPC for remote computation where client submits an integer value to the server and server calculates factorial and returns the result to the client program.

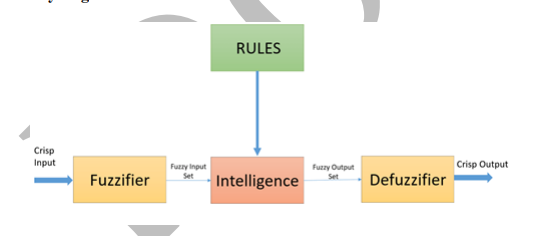
* **Outcome:** At end of this experiment, student will be able understand FUZZY w.r.t SET
* **Hardware Requirement:** Computer System, Linux(Ubantu)
* **Software Requirement:** PyCharm IDE

**Theory:**

Logic is a tool for reasoning propositions that can be manipulated with mathematical precepts. Fuzzy Logic: The word fuzzy means uncertainty. Any particular event which do not result any of the exact value (i.e. true or false) is fuzzy. A proposition is a declarative or linguistic statement within a universe of discourse. For example, Elizabeth is tall. In classical logic a proposition is either true or false. That means the proposition ‘Elizabeth is tall’ can be either true or false.

**shows a real-world situation where the glass is more than half full of water.**

Fuzzy logic is a transition from absolute truth to partial truth. That is, from a variable x (True or False) to a linguistic variable ‘Almost full’, ‘Very close to empty’, etc. From this perspective, fuzzy logic can be seen as a reasoning formalism of humans where all truthsare partial or approximate and any falseness is represented by partial truth

****

**Rule Base:** It contains all the rules and the if-then conditions offered by the experts to control the decision-making system.

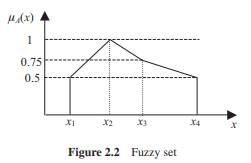
**Inference Engine:** It helps you to determines the degree of match between fuzzy input and the rules. Based on the % match, it determines which rules need implement according to the given input field. After this, the applied rules are combined to develop the control actions.

**Fuzzification** step helps to convert inputs. It allows you to convert, crisp numbers into fuzzy sets. Crisp inputs measured by sensors and passed into the control system for further processing. Like Room temperature, pressure, etc.

**Defuzzification:** At last the Defuzzification process is performed to convert the fuzzy sets into a crisp value. There are many types of techniques available, so you need to select it which is best suited when it is used with an expert system.

**For example**, For example, a fuzzy set A = {x1, x2, x3, x4} in X is characterized by the membership function μA(x) which maps each point x in X to real values 0.5, 1, 0.75 and 0.5. μA(x) represents the degree of membership of x in A and the mapping is only limited by μA(x) ∈ [0, 1]. In classical set theory, the membership function can take only two values: 0 and 1, i.e., either μA(x)=1 or μA(x) = 0. In set-theoretic notation this is written as μA(x) ∈ {0, 1}. A fuzzy set is an extension of a classical set. If X is the universe of discourse and its elements are denoted by x, then a fuzzy set A in X is defined as a set of ordered pairs

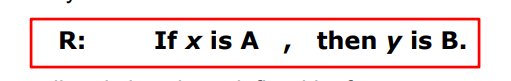
A = {x, μA(x) | x ∈ X}



This mapping can be depicted pictorially, as shown in Figure 2.2**. In Figure 2.2,** x1, x2, x3 and x4 have membership grades of 0.5, 1, 0.75 and 0.5, respectively, written as μA (x1) = 0.5, μA (x2) = 1, μA (x3) = 0.75 and μA (x4) = 0.5. A notational

**convention of fuzzy sets** for a discrete and finite universe of discourse X in practice is written as A = {μA(x1)/x1 + μA(x2)/x2 +···+ μA(xn)/xn} = n i=1 μA(xi)/xi

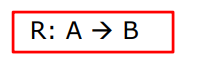
**Fuzzy Rules:**

****

where A and B are linguistic values defined by fuzzy sets on universes of discourse X and Y, respectively.

The rule is also called a “fuzzy implication” or fuzzy conditional statement. The part “x is A” is called the “antecedent”, while “y is B” is called the “consequence” or “conclusion”. It typically expresses an inference such that if we know a fact or a hypothesis (antecedent,), then we can infer, or derive, another fact called a conclusion (consequent)..

Before we employ fuzzy if-then rules to model and analyse a system, first we have to formalize what is meant by the expression: R: If x is A then y is B which is sometimes abbreviated as



**Conclusion: -** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-------------------------------------------------------------------------------------------------------------------------

**Questions:**

Q1. What FUZZY?explain with Ex.?

Q2. Fuzzy Application ?

Q3. Explain Fuzzy Architecture?

Q4. Differentiate Classical Vs Fuzzy Rule?

**Assignment No. 5 (Part II)**

**Aim**: Design a distributed application using RPC for remote computation where client submits an integer value to the server and server calculates factorial and returns the result to the client program.

* **Outcome:** At end of this experiment, student will be able understand Swam Intelligence
* **Hardware Requirement:** Computer System, Linux (Ubantu)
* **Software Requirement:** PyCharm IDE

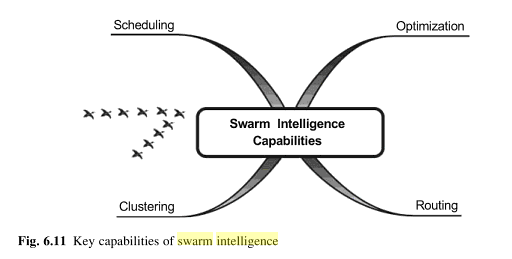
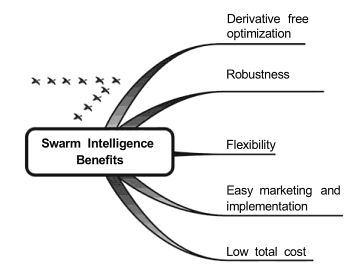
**Theory:**

**Swarm Intelligence (S.I.)** was introduced by **Gerardo Beni and Jing Wang in the year 1989**. S.I. simply means using the knowledge of collective objects (people, insects, etc.) together and then reaching the optimized solution for a given problem. **“Swarm” means a group of objects (people, insects**,**etc.)**. In other words, let’s say we give a problem statement to a single person and tell him or her to go through this problem and then give the solution, then this means that we will consider the solution of that particular person only, but the problem is that the solution given by that person may not be the best solution or maybe, that solution is not good for others. So to avoid that, what we do is we give that problem to a certain amount of people together (swarm) and ask them to reach the best solution possible for that problem, and then computing all the responses together to reach the best solution possible, so here we are using the knowledge of the group as a whole to reach to the best solution or optimized solution for that problem and that solution will be good for all of them individually too, so that is the idea behind swarm intelligence.

### **Example**

Let’s say we have a jar containing 500 marbles in that. The question is without touching the jar a person needs to predict how many marbles are in that jar. Suppose we take only one response from a person and it predicts that according to him the jar contains 400 marbles. So by this result, we can conclude that this estimation of that person is not very bad since the **difference (error) is of 100** only, but this might not be the best solution, we can optimize this even more. So now what we will do is instead of taking response from only one person we will be taking response from 10 people let’s say. Let ‘P’ denote a person therefore the responses are as follows:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 |
| 400 | 450 | 550 | 600 | 480 | 390 | 520 | 490 | 510 | 450 |

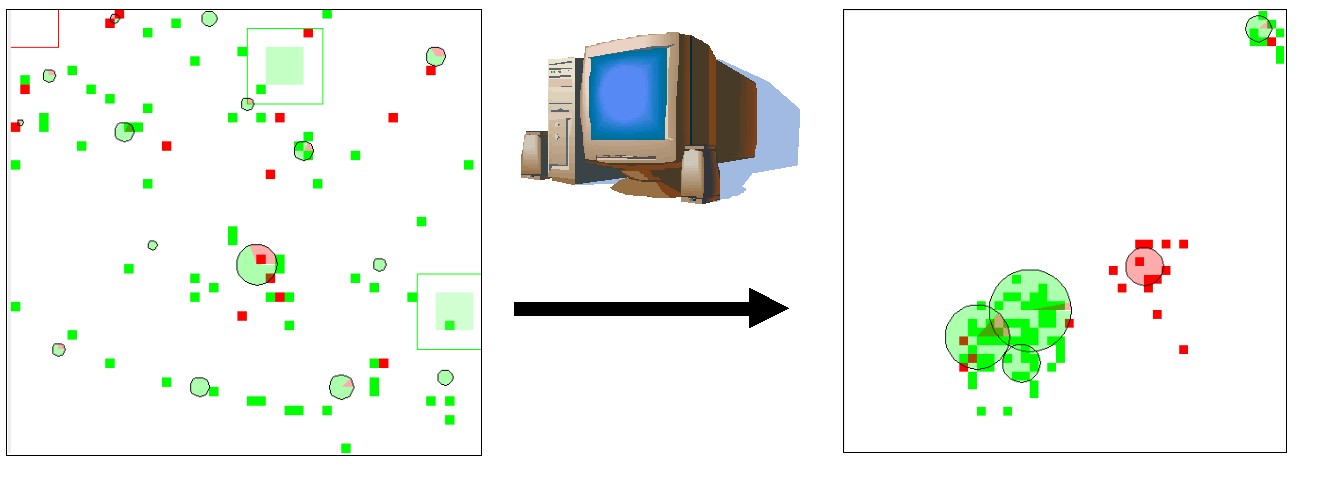


|  |
| --- |
| [Swarm Intelligence Capabilities:](https://www.techferry.com/articles/swarm-intelligence.html) |
| 1. [Scheduling / Load Balancing:](http://homes.ieu.edu.tr/~agokce/Courses/Swarm%20Intelligence%20Intro.pdf)   The emphasis is on he elative position of the job rather than its direct predecessor or its direct successor in the schedule and summation evaluation rule / global pheromone evaluation rule is followed. |

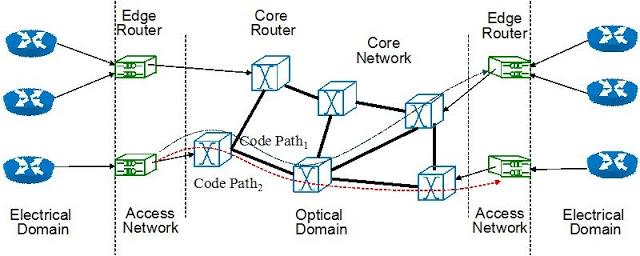
A diagram of a plant growing

Description automatically generated

1. **Clustering:**A cluster is a collection of agents which are similar and are dissimilar to the agents in other clusters.



**Optimization:** An optimization problem is the problem of finding the Best Solution / Minimal Cost Solution from all the feasible solutions.



1. **Routing:**This is based on the principle that backward ants utilize the useful information gathered by the forward ants on their trip from source to destination.

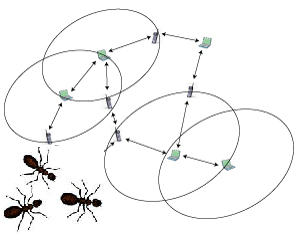


Fig: [The AntHocNet](http://www.idsia.ch/~gianni/AntHocNet/anthocnet.html) routing algorithm for MANETs (mobile ad hoc networks)

|  |
| --- |
| [**Advantages:**](https://www.techferry.com/articles/swarm-intelligence.html) |
| 1) **Flexible:** The colony respond to internal disturbances and external challenges. |
| 2) **Robust:**Tasks are completed even if some agents fail. |
| 3) **Scalable:**From a few agents to millions |
| 4) **Decentralized:**There is no central control in the colony. |
| 5) **Self-organized:**The solutions are emergent rather than pre-defined. |
| 6) **Adaptation:**The swarm system can not only adjust to predetermined stimuli but also to new stimuli. |
| 7) **Speed:**Changes in the network can be propagated very fast. |
| 8) **Modularity:** Agents act independently of other network layers. |
| 9) **Parallelism:** Agents' operations are inherently parallel. |

**Conclusion: -** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-------------------------------------------------------------------------------------------------------------------------

**Questions:**

Q1. What is Swarm Intelligence ?

Q2. What Are the Capabilities of Swarm Intelligence?

Q3. Advantages Of Swarm?

Q4. Explain Clustering with Swarm?

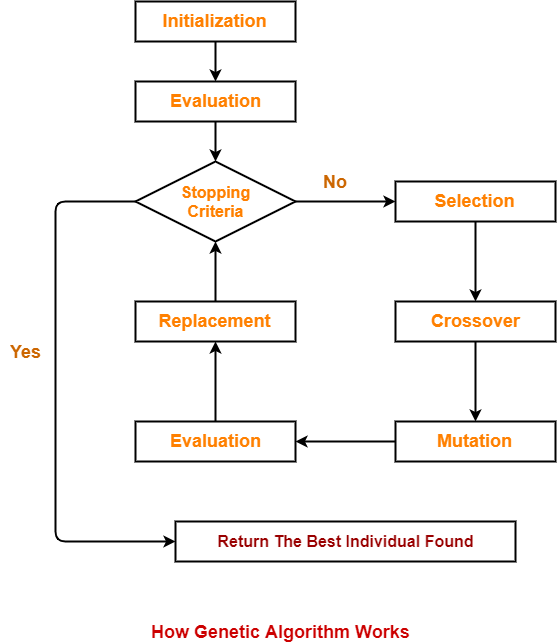
**Assignment No. 6 (Part 1)**

**Aim**: Optimization of genetic algorithm parameter in hybrid genetic algorithm-neural network modelling: Application to spray drying of coconut milk.

* **Outcome:** At end of this experiment, student will be able understand Genetic Algorithm
* **Hardware Requirement:** Computer System, Linux(Ubantu)
* **Software Requirement:** PyCharm IDE

**Theory:**

**Genetic Algorithm:**

****

* Abstraction of real biological evolution
* Solve complex Problems(NP-Hard type ex.TSP)
* Focus On Optimaization
* Population of possible solution for a given problem
* From group of individuals the best one Will survive

Genetic Algorithms offer the following advantages-

Point-01:

 Genetic Algorithms are better than conventional AI.

* This is because they are more robust.

 Point-02:

 They do not break easily unlike older AI systems.

* They do not break easily even in the presence of reasonable noise or if the inputs get change slightly.

Point-03:

While performing search in multi modal state-space or large state-space,

Genetic algorithms has significant benefits over other typical search optimization techniques.

* GA (Genetic Algorithm) is good at taking larger, potentially huge search space and navigating them looking for optimal solution which we might not find in lifetime.
* GA is better than other traditional algorithm in that they are more robust.
* They do not break easily even if the inputs are changed slightly or in the presence of reasonable noise.

GA is used to resolve complicated optimization problems, such as , organizing the time table, scheduling job shop, playing games.

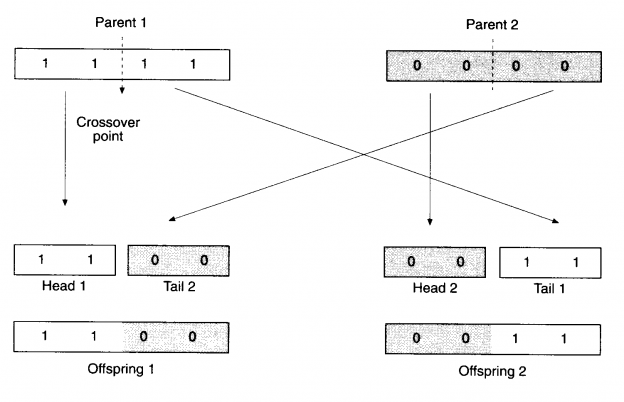
The concept of GA is directly derived from natural evolution and heredity i.e. inheritance, where child inherits the characters (stored in the chromosomes) from the parent.

**Operators in GA:**  
***1.Crossover (Recombination):-***

**Crossover** is the process of **taking two parent solutions and producing from them a child**. After the selection (reproduction) process, the population is enriched with better individuals. Crossover operator is applied to the mating pool with the hope that it creates a better offspring.

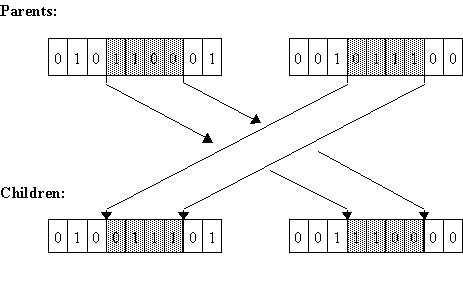
**The various crossover techniques are-**

*i).****Single-Point Crossover****-*Here the two mating chromosomes are cut once at corresponding points and the sections after the cuts exchanged.



***ii). Two-Point Crossover-***

Heretwo crossover points are chosen and the contents between these points are exchanged between two mated parents.

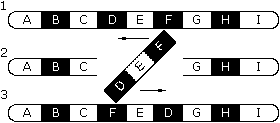


***2. Inversion:-***

Inversion operator inverts the bits between two random sites.

 01  0011  1

Then, 0111001



***3. Deletion:-***

***i).Deletion and duplication****-*Here any two or three bits in random are selected and their previous bits are duplicated.

before duplication: 00   1001  0

deletion: 00  10\_  \_   0

duplication: 00  1010  0

***ii). Deletion and regeneration****-*Here bits between the cross site are deleted and regenerated randomly.

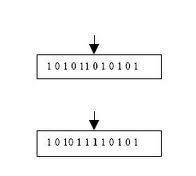
10  0110 1

10 \_ \_ \_ \_ 1

10  1101  1

***4. Mutation:-***

After crossover, the strings are subjected to mutation. Mutation prevents the algorithm to be trapped in a local minimum. It plays the role of recovering the genetic materials as well as for randomly distributing genetic information. It helps escape from local minima’s trap and maintain diversity in the population. Mutation of a bit involves flipping a bit, changing 0 to 1and vice-versa.



**Conclusion: -** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-------------------------------------------------------------------------------------------------------------------------

**Questions:**

Q1. Define Phenotype & Genotype with Ex.?

Q2. Define Encoding & Decoding And Explain Its Technique?

Q3. Define Term Population,Genes,Fitness Function w.r.t Genetic Algorithm?

Q4. Explain Genetic algorithm with its architecture?

**Assignment No. 7 (Part 1)**

**Aim**: Implement DEAP (Distributed Evolutionary Algorithms) using Python

* **Outcome:** At end of this experiment, student will be able understand DEAP
* **Hardware Requirement:** Computer System, Linux(Ubuntu)
* **Software Requirement:** PyCharm IDE

**Theory:**

* DEAP is a novel evolutionary computation framework for rapid prototyping and testing of ideas. It seeks to make algorithms explicit and data structures transparent. It works in perfect harmony with parallelization mechanisms such as multiprocessing and [SCOOP](https://github.com/soravux/scoop).

DEAP includes the following features:

* Genetic algorithm using any imaginable representation
  + List, Array, Set, Dictionary, Tree, Numpy Array, etc.
* Genetic programming using prefix trees
  + Loosely typed, Strongly typed
  + Automatically defined functions
* Evolution strategies (including CMA-ES)
* Multi-objective optimisation (NSGA-II, NSGA-III, SPEA2, MO-CMA-ES)
* Co-evolution (cooperative and competitive) of multiple populations
* Parallelization of the evaluations (and more)
* Hall of Fame of the best individuals that lived in the population
* Checkpoints that take snapshots of a system regularly
* Benchmarks module containing most common test functions
* Genealogy of an evolution (that is compatible with [NetworkX](https://github.com/networkx/networkx))
* Examples of alternative algorithms : Particle Swarm Optimization, Differential Evolution, Estimation of Distribution Algorithm

Overview

If you are used to any other evolutionary algorithm framework, you’ll notice we do things differently with DEAP. Instead of limiting you with predefined types, we provide ways of creating the appropriate ones. Instead of providing closed initializers, we enable you to customize them as you wish. Instead of suggesting unfit operators, we explicitly ask you to choose them wisely. Instead of implementing many sealed algorithms, we allow you to write the ones that fit all your needs. This tutorial will present a quick overview of what DEAP is all about along with what every DEAP program is made of.

**Types**

The first thing to do is to think of the appropriate type for your problem. Then, instead of looking in the list of available types, DEAP enables you to build your own. This is done with the [creator](https://deap.readthedocs.io/en/master/api/creator.html#module-deap.creator) module. Creating an appropriate type might seem overwhelming but the creator makes it very easy. In fact, this is usually done in a single line. For example, the following creates a FitnessMin class for a minimization problem and an Individual class that is derived from a list with a fitness attribute set to the just created fitness.

* **from** **deap** **import** base, creator
* creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
* creator.create("Individual", list, fitness=creator.FitnessMin)

That’s it. More on creating types can be found in the [Creating Types](https://deap.readthedocs.io/en/master/tutorials/basic/part1.html) tutorial.

**Initialization**

Once the types are created you need to fill them with sometimes random values or sometime guessed ones. Again, DEAP provides an easy mechanism to do just that. The [Toolbox](https://deap.readthedocs.io/en/master/api/base.html#deap.base.Toolbox) is a container for tools of all sorts including initializers that can do what is needed of them. The following takes on the last lines of code to create the initializers for individuals containing random floating point numbers and for a population that contains them.

* **import** **random**
* **from** **deap** **import** tools
* IND\_SIZE = 10
* toolbox = base.Toolbox()
* toolbox.register("attribute", random.random)
* toolbox.register("individual", tools.initRepeat, creator.Individual,
* toolbox.attribute, n=IND\_SIZE)
* toolbox.register("population", tools.initRepeat, list, toolbox.individual)
* This creates functions to initialize populations from individuals that are themselves initialized with random float numbers. The functions are registered in the toolbox with their default arguments under the given name. For example, it will be possible to call the function toolbox.population() to instantly create a population. More initialization methods are found in the [Creating Types](https://deap.readthedocs.io/en/master/tutorials/basic/part1.html) tutorial and the various [Examples](https://deap.readthedocs.io/en/master/examples/index.html).

**Operators**

* Operators are just like initializers, except that some are already implemented in the [tools](https://deap.readthedocs.io/en/master/api/tools.html#module-deap.tools) module. Once you’ve chosen the perfect ones, simply register them in the toolbox. In addition you must create your evaluation function. This is how it is done in DEAP.
* **def** evaluate(individual):
* **return** sum(individual),
* toolbox.register("mate", tools.cxTwoPoint)
* toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=1, indpb=0.1)
* toolbox.register("select", tools.selTournament, tournsize=3)
* toolbox.register("evaluate", evaluate)
* The registered functions are renamed by the toolbox, allowing generic algorithms that do not depend on operator names. Note also that fitness values must be iterable, that is why we return a tuple in the evaluate function. More on this in the [Operators and Algorithms](https://deap.readthedocs.io/en/master/tutorials/basic/part2.html) tutorial and [Examples](https://deap.readthedocs.io/en/master/examples/index.html). Algorithms Now that everything is ready, we can start to write our own algorithm. It is usually done in a main function. For the purpose of completeness we will develop the complete generational algorithm.
* **def** main():
* pop = toolbox.population(n=50)
* CXPB, MUTPB, NGEN = 0.5, 0.2, 40
* *# Evaluate the entire population*
* fitnesses = map(toolbox.evaluate, pop)
* **for** ind, fit **in** zip(pop, fitnesses):
* ind.fitness.values = fit
* **for** g **in** range(NGEN):
* *# Select the next generation individuals*
* offspring = toolbox.select(pop, len(pop))
* *# Clone the selected individuals*
* offspring = map(toolbox.clone, offspring)
* *# Apply crossover and mutation on the offspring*
* **for** child1, child2 **in** zip(offspring[::2], offspring[1::2]):
* **if** random.random() < CXPB:
* toolbox.mate(child1, child2)
* **del** child1.fitness.values
* **del** child2.fitness.values
* **for** mutant **in** offspring:
* **if** random.random() < MUTPB:
* toolbox.mutate(mutant)
* **del** mutant.fitness.values
* *# Evaluate the individuals with an invalid fitness*
* invalid\_ind = [ind **for** ind **in** offspring **if** **not** ind.fitness.valid]
* fitnesses = map(toolbox.evaluate, invalid\_ind)
* **for** ind, fit **in** zip(invalid\_ind, fitnesses):
* ind.fitness.values = fit
* *# The population is entirely replaced by the offspring*
* pop[:] = offspring
* **return** pop

It is also possible to use one of the four algorithms readily available in the [algorithms](https://deap.readthedocs.io/en/master/api/algo.html#module-deap.algorithms) module, or build from some building blocks called variations also available in this module.

# **Installation**

## Requirements

DEAP is compatible with Python 2.7 and 3.4 or higher. The computation distribution requires [SCOOP](http://www.pyscoop.org/). CMA-ES requires [Numpy](http://www.numpy.org/), and we recommend [matplotlib](http://www.matplotlib.org/) for visualization of results as it is fully compatible with DEAP’s API.

# **Install DEAP**

We encourage you to use [easy\_install](http://pythonhosted.org/distribute/easy_install.html) or [pip](http://www.pip-installer.org/en/latest/) to install DEAP on your system. Linux package managers like apt-get, yum, etc. usually provide an outdated version.

easy\_install deap

or

pip install deap

If you wish to build from sources, [download](https://pypi.python.org/pypi/deap/) or [clone](https://github.com/DEAP/deap.git) the repository and type:

python setup.py install

**Conclusion: -** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-------------------------------------------------------------------------------------------------------------------------

**Questions:**

Q1.What is Genetic algorithm with DEAP?

Q2.What is DEAP Framework?

Q3.Why Genetic algorithm used in optimization?

Q4.What is DEAP library in Python?

**Assignment No. 8 (Group A)**

**Aim**: .Implement Any one membership function by fuzzy & Calculate the Membership function values using triangular shape in Python

* **Outcome:** At end of this experiment, student will be able understand FUZZY Membership function and its Different visualization
* **Hardware Requirement:** Computer System, Linux(Ubantu)
* **Software Requirement:** PyCharm IDE, Jupyter IDE

**Theory:**

In fuzzy logic, membership functions are used to determine the degree of membership of an element in a fuzzy set. A membership function assigns a membership value between 0 and 1 to each element of the universe of discourse. This value represents the degree to which the element belongs to the fuzzy set.

There are various types of membership functions commonly used in fuzzy logic, including:

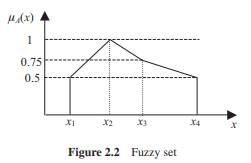
1. Triangular membership function: This is one of the simplest membership functions, defined by three parameters: the lower bound, the peak, and the upper bound.
2. Trapezoidal membership function: This function is similar to the triangular function but has an additional parameter defining the plateau between the lower and upper bounds.
3. Gaussian membership function: This function is bell-shaped and is defined by its mean and standard deviation.
4. Generalized bell-shaped membership function: This function is a more generalized form of the Gaussian function and is defined by three parameters: the center, the width, and the slope.
5. Sigmoidal membership function: This function is S-shaped and is often used to represent the transition between two states.

For example, For example, a fuzzy set A = {x1, x2, x3, x4} in X is characterized by the membership function μA(x) which maps each point x in X to real values 0.5, 1, 0.75 and 0.5. μA(x) represents the degree of membership of x in A and the mapping is only limited by μA(x) ∈ [0, 1]. In classical set theory, the membership function can take only two values: 0 and 1, i.e., either μA(x)=1 or μA(x) = 0. In set-theoretic notation this is written as μA(x) ∈ {0, 1}.

A fuzzy set is an extension of a classical set. If X is the universe of discourse and its elements are denoted by x, then a fuzzy set A in X is defined as a set of ordered pairs

A = {x, μA(x) | x ∈ X}

This mapping can be depicted pictorially, as shown in Figure 2.2**. In Figure 2.2,** x1, x2, x3 and x4 have membership grades of 0.5, 1, 0.75 and 0.5, respectively, written as μA (x1) = 0.5, μA (x2) = 1, μA (x3) = 0.75 and μA (x4) = 0.5. A notational



Membership Functions

Very often, real-world situations are not certain and cannot be described precisely. For example, the uncertainty in Example is belonging to Tall or very Tall. The uncertainties of expressions like ‘very nice’, ‘too small’, ‘high value’ are called fuzziness. The function that characterizes the fuzziness of a fuzzy set A in X, which associates each point in X with a real number in

the interval [0, 1], is called a membership function (MF).

There is no strict rule for defining a membership function.

The choice of membership function is usually problem-dependent and often determined heuristically and subjectively.

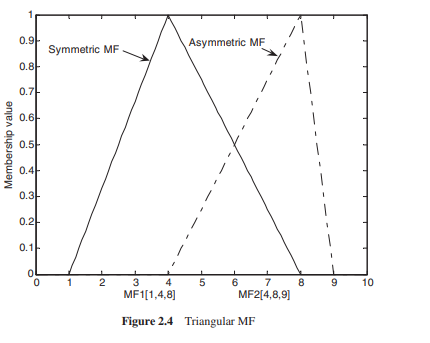
**Most widely used MFs in the fuzzy logic literature are**

**1)Triangular**

**2)Trapezoidal**

**3)Gaussian and**

**4)bell-shaped functions**



\*\* Triangular MF

A triangular MF is specified by three parameters {a, b, c}, shown in Figure 2.4(Previous slide) and defined

The parameters {a, b, c} with a < b < c determine the x coordinates of the three corners of the underlying triangular MF.

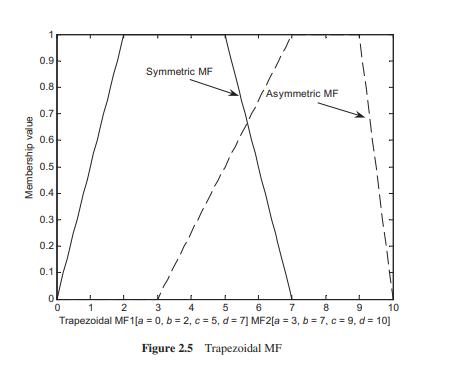
**Triangular MFs can be asymmetric**, depending on the relations a ≤ b and b ≤ c.

Figure 2.4 shows a symmetric and an asymmetric triangular MF.

Trapezoidal MF (Quadrilateral)

**A trapezoidal MF is specified by four parameters{a, b, c, d}, shown in Figure 2.5 and defined as**

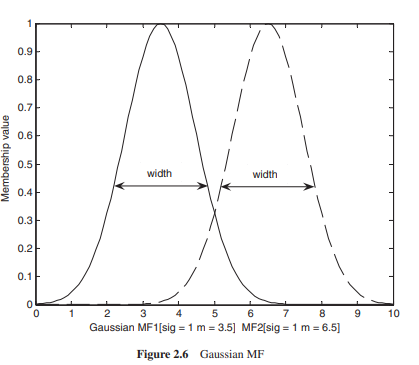
The parameters {a, b, c, d} with a < b < c < d determine the x coordinates of the four corners of the underlying trapezoidal MF. **Trapezoidal MFs can be asymmetric**, depending on the relations a ≤ b and c ≤ d**. Both triangular and trapezoidal MFs can be symmetric or asymmetric, which is seen as an advantage for some applications**. Owing to their simple formulae and computational efficiency, **both triangular and trapezoidal MFs have been used extensively, especially in online applications**



Gaussian MF(Curve Shape):

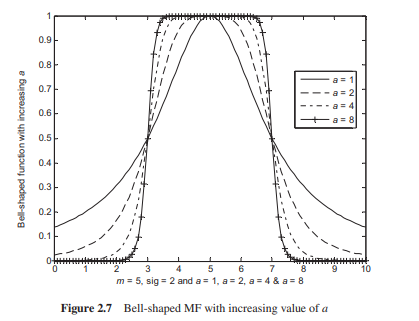
A Gaussian MF is specified by two parameters {m, σ}, shown in Figure 2.6 and defined as

The parameters m and σ represent the centre and width of the Gaussian MF, respectively



Bell-shaped MF:

A bell-shaped MF is specified by three parameters{m, σ, a}, shown in Figure 2.7 and defined as **The parameters m and σ represent the centre and width of the bell-shaped MF, respectively**. Parameter a, usually positive, controls the slope of the MF as shown in Figure 2.7. The MF is narrower with increasing value of a



**Conclusion: -** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-------------------------------------------------------------------------------------------------------------------------

**Questions:**

Q1. What FUZZY?Define Membership function?

Q2. Enlist & Explain Different Membership types briefly? ?

Q3. Write Short Note on triangular Membership function?

Q4. Discuss real time example where these all membership function used currently?

**Assignment No. 9 (Part II)**

**Aim**: **To apply the artificial immune pattern recognition to perform a task of structure damage Classification**

**Outcome:** At end of this experiment, student will be able understand AIPR

* **Hardware Requirement:** Computer System, Linux(Ubuntu)
* **Software Requirement:**  PyCharm IDE OR, Jupyter

**Theory:**

The task at hand is to classify structural damage in various infrastructure components such as buildings, bridges, roads, etc. Given data representing different types and degrees of structural damage, the goal is to develop a classification system that can accurately identify and classify the severity of damage.

Approach using Artificial Immune Pattern Recognition (AIPR):

1. Understanding AIPR:

* AIPR is inspired by the human immune system's ability to recognize and respond to foreign pathogens.
* In AIPR, the concept of antigens and antibodies is used to detect patterns in data.
* Antigens represent patterns in the data, while antibodies are generated to recognize and classify these patterns.
* AIPR algorithms evolve a set of antibodies through a process of affinity maturation and selection to achieve accurate pattern recognition.
* 2. Data Preprocessing:
* Begin by collecting and preprocessing the data representing structural damage.
* Preprocessing may involve tasks such as normalization, feature extraction, and dimensionality reduction.

3. Representation of Data:

* Represent the structural damage data as antigens, where each antigen represents a specific pattern or feature in the data.
* Antigens can be represented as vectors or matrices depending on the nature of the data.

4. Generation of Antibodies:

* Initialize a set of antibodies representing potential classifiers.
* Antibodies are initialized randomly or using a predefined strategy.

5. Affinity Maturation:

* Apply affinity maturation to evolve the antibodies to better recognize antigens.
* During affinity maturation, antibodies undergo mutation and selection to improve their affinity towards antigens.

6. Training:

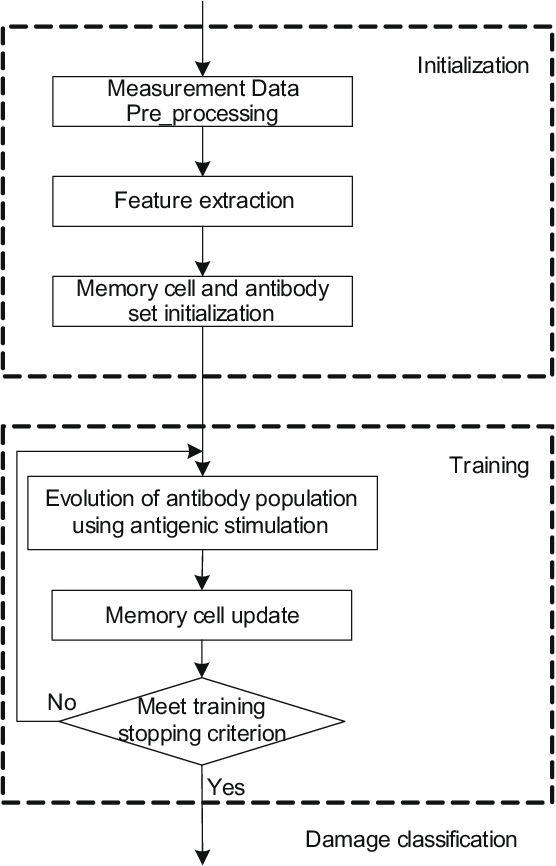
* Train the AIPR system using the preprocessed data.
* During training, the antibodies are exposed to antigens, and their affinities are adjusted based on the similarity between antibodies and antigens.

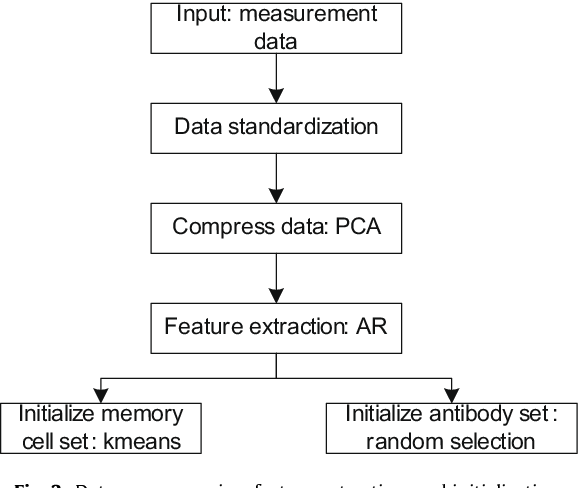
7. Classification:

* After training, the evolved antibodies can be used for classification.
* Given a new instance of structural damage, the system can classify its severity based on the affinity of antibodies towards the patterns present in the instance.

8. Evaluation:

* Evaluate the performance of the AIPR system using metrics such as accuracy, precision, recall, and F1-score.
* Fine-tune the system parameters and repeat the training and evaluation process as necessary to improve performance.





**Data preprocessing** in Artificial Immune Pattern Recognition (AIPR) plays a crucial role in preparing the raw data for effective pattern recognition. Here's how data preprocessing can be conducted in the context of AIPR for the task of structure damage classification:

1. **Data Cleaning:**
   * Remove any noise or irrelevant information from the dataset.
   * Handle missing values by imputation or removal.
2. **Normalization:**
   * Scale the features to a similar range to ensure that they contribute equally to pattern recognition.
   * Common normalization techniques include Min-Max scaling and Z-score normalization.
3. **Feature Extraction:**
   * Extract relevant features from the raw data that capture important characteristics related to structural damage.
   * Feature extraction techniques can include statistical measures, frequency domain analysis, or domain-specific features.
4. **Dimensionality Reduction:**
   * Reduce the dimensionality of the feature space to mitigate the curse of dimensionality and improve computational efficiency.
   * Techniques such as Principal Component Analysis (PCA) or feature selection methods can be employed for dimensionality reduction.
5. **Encoding Categorical Variables:**
   * If the dataset contains categorical variables, encode them into numerical values using techniques such as one-hot encoding or label encoding.
6. **Data Balancing (if applicable):**
   * If the dataset is imbalanced, where certain classes of structural damage are underrepresented, apply techniques such as oversampling, undersampling, or synthetic data generation to balance the dataset.
7. **Data Partitioning:**
   * Split the preprocessed dataset into training, validation, and test sets for model development, evaluation, and testing purposes, respectively.
8. **Data Augmentation (optional):**
   * Augment the dataset by generating additional samples through transformations such as rotation, translation, or scaling.
   * Data augmentation can help improve model generalization and robustness, especially when dealing with limited data.
9. **Data Representation:**
   * Represent the preprocessed data in a suitable format for AIPR algorithms to process.
   * Data representation may involve converting the data into antigenic form, where each instance is represented as an antigen with specific features.

**Data Representaion: Antigen Representation:**

* In AIPR, antigens represent patterns or instances from the dataset.
* Each antigen encapsulates the features or attributes of a data point.
* Antigens can be represented as vectors, matrices, or any other suitable data structure depending on the nature of the data.
* For structure damage classification, antigens may include various features such as material properties, geometric characteristics, environmental conditions, etc.

**Classification:**

* Given a new antigen (representing a new instance of structural damage), classify it by evaluating its similarity to the antibodies.
* Measure the similarity between the antigen and each antibody using a predefined similarity metric (e.g., Euclidean distance or cosine similarity).
* Classify the antigen based on the classification of the antibody with the highest affinity.
* The classification result may include the predicted class label and the confidence level (affinity) associated with the classification.

**Conclusion: -** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-------------------------------------------------------------------------------------------------------------------------

**Questions:**

Q1. Describe AIS?

Q2. Explain Data Representation in AIS ?

Q3. Define Data Cleaning & Feature Extraction?

Q4. Why their is need of AIS?

**Assignment No. 10 (Part I)**

**Aim**: **Write code to simulate requests coming from clients and distribute them among the servers using the load balancing algorithms.**

**Outcome:** At end of this experiment, student will be able understand how to manage Load of Servers by balancing

* **Hardware Requirement:** Computer System, Linux(Ubuntu)
* **Software Requirement:**  PyCharm IDE OR, Jupyter,Google Colab

**Theory:**

**Introduction:**Load balancing is a crucial technique in distributed computing that ensures efficient utilization of resources by distributing incoming requests across multiple servers. It helps in preventing server overload, reducing response time, and improving system performance.

In this experiment, we implement two common load balancing algorithms:

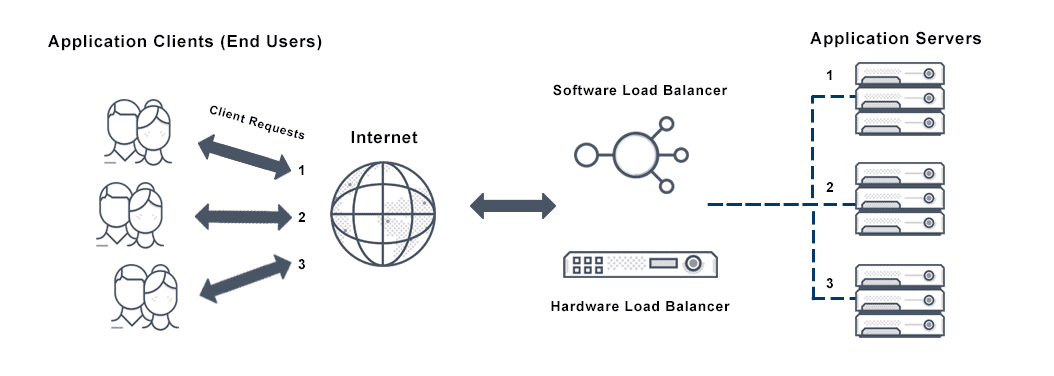
1. **Round Robin Load Balancing**
2. **Least Connections Load Balancing**

**Load Balancing Algorithms:**

1. **Round Robin Algorithm:**
   * Requests are assigned to servers in a cyclic order.
   * Each server gets an equal number of requests in sequence.
   * Simple to implement and effective for evenly distributed loads

**How It Works: Source** [**View**](https://www.vmware.com/topics/round-robin-load-balancing)

Consider a scenario with three servers: Server A, Server B, and Server C. The Round Robin algorithm assigns the first incoming request to Server A, the second to Server B, the third to Server C, and then cycles back to Server A for the fourth request. This pattern continues, rotating through the list of servers, thereby distributing the requests evenly.



**Advantages:**

* **Simplicity:** The algorithm is easy to implement and requires minimal computational overhead.
* **Fairness:** Each server receives an approximately equal number of requests, preventing any single server from becoming a bottleneck.

**Limitations:**

While Round Robin Load Balancing is effective in homogeneous environments where servers have similar capacities and the incoming requests are relatively uniform in resource demands, it has certain limitations:

* **Ignoring Server Load:** The algorithm does not consider the current load or performance capacity of each server. In cases where servers have varying processing powers or some requests require more resources than others, this can lead to inefficient load distribution.  
   [f5.com](https://www.f5.com/glossary/round-robin-load-balancing?utm_source=chatgpt.com)
* **Lack of Fault Tolerance:** If a server becomes unavailable or unresponsive, the standard Round Robin algorithm will continue to route requests to it, leading to potential failures or delays.

**Variations:**

To address some of these limitations, variations of the Round Robin algorithm have been developed:

* **Weighted Round Robin:** This approach assigns a weight to each server based on its capacity or performance metrics. Servers with higher weights receive a proportionally larger number of requests, allowing for better load distribution in heterogeneous environments.  
   [geeksforgeeks.org](https://www.geeksforgeeks.org/load-balancing-algorithms/?utm_source=chatgpt.com)

**Use Cases:**

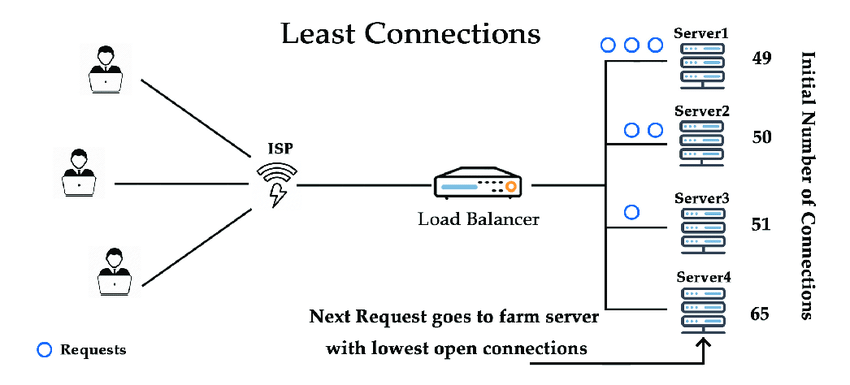
Round Robin Load Balancing is particularly useful in scenarios where:

* Servers have similar specifications and capabilities.
* The workload is evenly distributed, with requests requiring comparable processing resources.
* A simple and easy-to-implement load balancing solution is desired.

1. **Least Connections Algorithm: Source** [**View**](https://docs.netscaler.com/en-us/citrix-adc/current-release/load-balancing/load-balancing-customizing-algorithms/leastconnection-method.html)
   * Requests are assigned to the server with the fewest active connections.
   * Ensures that no single server gets overloaded if the requests are of varying durations.
   * More dynamic and efficient in scenarios where processing times differ.

**How It Works:**

1. **Connection Monitoring:** The load balancer continuously tracks the number of active connections for each server in the pool.
2. **Request Allocation:** When a new request arrives, the load balancer evaluates the active connection count for each server and assigns the request to the server with the least number of active connections at that moment.



**Advantages:**

* **Dynamic Load Distribution:** By considering the current number of active connections, the algorithm adapts to fluctuating workloads, distributing requests based on real-time server load.
* **Improved Resource Utilization:** Ensures that all servers operate efficiently, reducing the likelihood of some servers being overburdened while others remain underutilized.

**Limitations:**

* **Server Capacity Ignorance:** The basic Least Connections algorithm does not account for differences in server capabilities. For instance, a high-capacity server and a low-capacity server are treated equally, which might lead to suboptimal performance.
* **Connection Persistence:** In scenarios where connections have long durations, a server might accumulate numerous connections, potentially leading to uneven load distribution over time.

**Variations:**

* **Weighted Least Connections:** To address differences in server capacities, this variation assigns a weight to each server based on its processing power or other performance metrics. Servers with higher weights receive a proportionally larger share of the incoming requests, balancing the load according to server capabilities.

**Use Cases:**

* **Heterogeneous Server Environments:** Ideal for setups where servers have varying processing powers, ensuring that more capable servers handle a larger portion of the load.
* **Variable Request Processing Times:** Suitable for applications where the duration or resource consumption of requests is unpredictable, as the algorithm dynamically adjusts to the current load on each server.

**Implementation Details:**

* Each server maintains an active connection count.
* The Round Robin algorithm assigns requests sequentially.
* The Least Connections algorithm assigns requests to the least loaded server.
* Requests are processed and then released to simulate real-world behavior.
* The system prints request allocation and server status after execution.

**Expected Output:**

* The Round Robin algorithm should distribute requests evenly across all servers.
* The Least Connections algorithm should distribute requests based on the server load, leading to a more balanced request handling process.
* The program should show step-by-step request assignments and the final load distribution.

**Conclusion: -** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-------------------------------------------------------------------------------------------------------------------------

**Questions:**

1. What is load balancing and why is it important?
2. Explain the working principle of the Round Robin algorithm.
3. How does the Least Connections algorithm dynamically allocate requests?
4. Which load balancing algorithm is best suited for real-time applications and why?
5. What are other load balancing techniques used in modern distributed systems?