A Project
Report on

Cloud Task Scheduling using Machine Learning Algorithm

carried out as part of Minor project CC1634

Submitted by

RAJAT SAXENA
199303182
SANDESH MURDIA
199303072

6th Semester
of
BACHELOR OF TECHNOLOGY
In Computer and Communication

UNDER GUIDANCE OF
DR. PUNIT GUPTA
(Faculty Supervisor)

MANIPAL UNIVERSITY JAIPUR

INSPIRED BY LIFE

Department of Computer & Communication Engineering,

School of Computing and IT,

Manipal University Jaipur, *May 2022*

DECLARATION

I hereby declare that the project entitled "***Cloud Task Scheduling using Machine Learning algorithms***" submitted as part of the partial course requirements for the course ***Minor Project (CC1634),*** for the award of the degree of Bachelor of Technology in Computer & Communication Engineering at Manipal University Jaipur during the ***May 2022, 6th*** semester, has been carried out by me. I declare that the project has not formed the basis for the award of any degree, associate ship, fellowship or any other similar titles elsewhere.

Further, I declare that I will not share, re-submit or publish the code, idea, framework and/or any publication that may arise out of this work for academic or profit purposes without obtaining the prior written consent of the Course Faculty Mentor and Course Instructor.

Signature of the Student 1:

Signature of the Student 2:

Place: Jaipur

Date: 21-05-2022

CERTIFICATE

This is to certify that the project entitled "***Cloud Task Scheduling using Machine Learning algorithms***" is a bonafide work carried out as part of the course ***Minor Project (CC1634)***, under my guidance by Sandesh Murdia and Rajat Saxena, student of ***BTech. CCE Semester 6th*** at the Department of Computer & Communication Engineering, Manipal University Jaipur, during the academic semester ***_6th,*** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer & Communication Engineering, at MUJ, Jaipur.

Place: Jaipur

Date: 21-05-2022 Signature of the Instructor (s)

Table Of Contents

INTRODUCTION

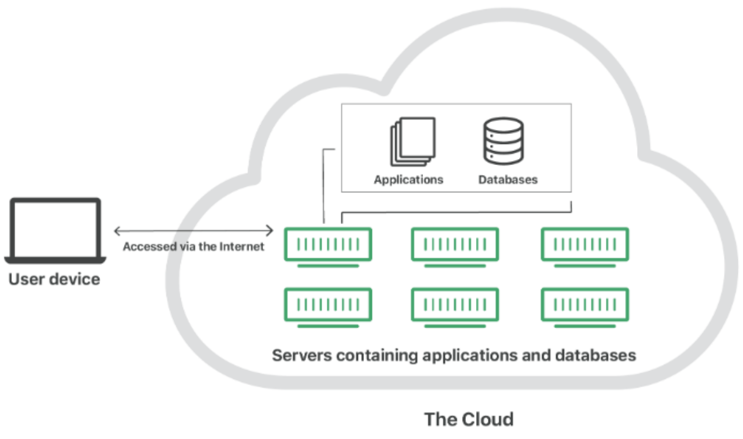
What is Cloud?

Cloud refers to software or we can say services which runs on the Internet, rather than locally on your computer. Examples- Google Drive, Apple iCloud, Microsoft OneDrive etc.

The cloud enables users to access the same files and applications from almost any device in the world, because the everything in cloud takes place on servers in a data centre, instead of locally on the user device. This is why a we can log our google account on a new phone after our old phone breaks and still find the account which we used to use in our old phone in place, with all our data, photos, videos, and conversation history. It works the same way with cloud email providers like Gmail or Microsoft Office 365, and with cloud storage providers like Dropbox or Google Drive.

When we think what IT really needs, and that is a way to increase capacity or add features without investing on new infrastructure, training new personnel, or licensing new software.

Cloud computing, grid computing and high performance computing and data centre computing all belong to parallel computing. There are almost 20 definitions of cloud computing that seem to only focus on certain aspects of this technology. Many big tech giants are jumping into the pool of cloud computing because of the economic pattern and also distinguishes from other research area such as hpc and grid computing. Companies like Microsoft, amazon, apple amazon, IBM will have a full fledged war on cloud computing in the upcoming future.


Applications
Databases
User device
Accessed via the Internet
Servers containing applications and databases
The Cloud

Basic Framework of Cloud

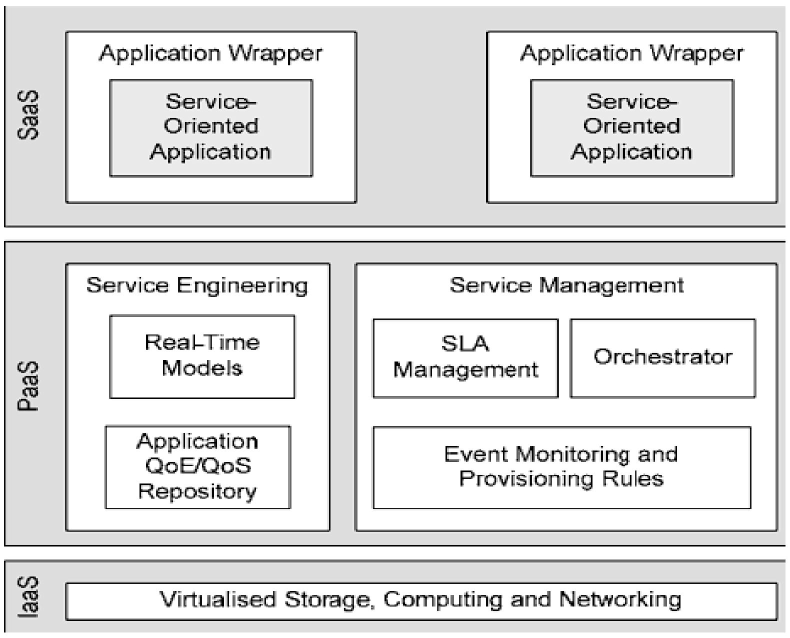
Service models of Cloud Computing-

Software-as-a-Service (SaaS): Instead of users installing an application on their device, SaaS applications are hosted on cloud servers, and users access them over the Internet. SaaS is like renting a house: the landlord maintains the house, but the tenant mostly gets to use it as if they owned it. Examples of SaaS applications include Salesforce, Mail-Chimp, and Slack. Software as a service is a software delivery model ,which provides users to access business remotely as a service. In this service the cost of install all the infrastructure is included in a single software which charges the user on monthly or yearly basis and helps in cost cutting. These services basically provide services with specific functionalities . most of the SaaS service subscribers, especially those medium to large companies , have certain applications already deployed on their premises. By doing this their application environment becomes hybrid . In SaaS every application has its own application interface and related access control ,thus users must should switch among different user identity and passwords information required by SaaS.

Platform-as-a-Service (PaaS): In this model, companies don't pay for hosted applications; instead, they pay for the things they need to build their own applications. PaaS vendors offer everything necessary for building an application, including development tools, infrastructure, and operating systems, over the Internet. PaaS can be compared to renting all the tools and equipment necessary for building a house, instead of renting the house itself. PaaS examples include Heroku and Microsoft Azure. Soft real time applications are traditionally developed without any real time methodology or run time support from the infrastructure on which they run . it directly results in purchasing of expensive software and hard drives. And also there is no guarantee or control on the behaviour of the application as a result. For such application PaaS is needed to support techniques for modelling, predicting ,provisioning and monitoring resources .

A PaaS structure requires the following key features:

- **Real time Qos specification**
- **Event prediction**
- **Dynamic SLA negotiation**
- **On- demand resource provisioning**
- **Qos event monitoring**


SaaS
Application Wrapper
Service-Oriented Application
Application Wrapper
Service-Oriented Application
PaaS
Service Engineering
Real-Time Models
Application QoE/QoS Repository
Service Management
SLA Management
Orchestrator
Event Monitoring and Provisioning Rules
IaaS
Virtualised Storage, Computing and Networking

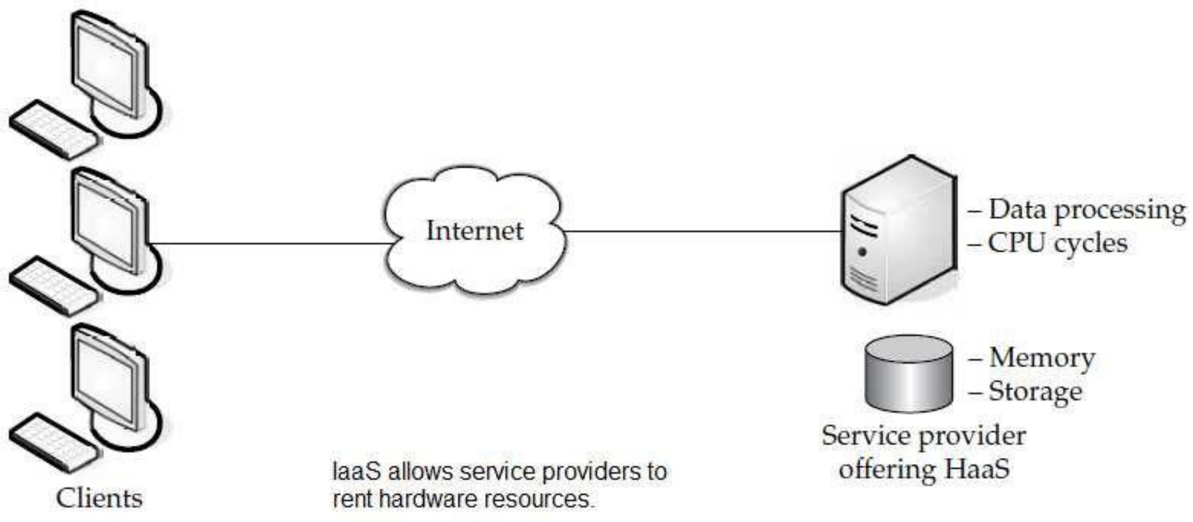
<u>Platform as a service functioning</u>

Infrastructure-as-a-Service (IaaS): In this model, a company rents the servers and storage they need from a cloud provider. They then use that cloud infrastructure to build their applications. IaaS is like a company leasing a plot of land on which they can build whatever they want — but they need to provide their own building equipment and materials. IaaS providers include Digital-Ocean, Google Compute Engine, and OpenStack. IaaS provides access to shared resources on a need basis, without revealing details like location and hardware to clients, and also it provides details like server images on demand, storage, queueing. The major issues that are commonly associated with IaaS in cloud systems are resource management, network infrastructure management, virtualization and multi-tenancy, data management, application management interfaces, interoperability, etc.

The major issues in IaaS are:

- Virtualization and multi tenancy.
- Resource management.
- Network infrastructure management.
- Security, privacy and compliance.
- Data management.

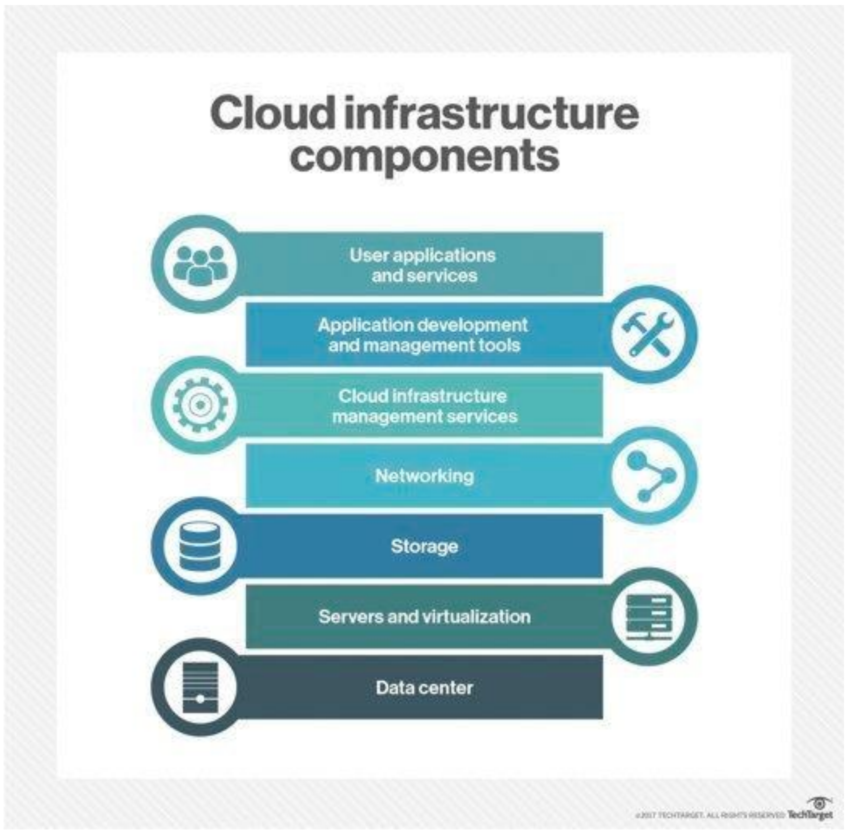
IaaS in general offers a combination of hosting, hardware provisioning and basic services needed to run a cloud.


Internet
– Data processing
– CPU cycles
– Memory
– Storage
Service provider offering HaaS
IaaS allows service providers to rent hardware resources.
Clients

Cloud Infrastructure-

It is a term used to describe the components needed for cloud computing, which includes hardware, abstracted resources, storage, and network resources. Think of cloud infrastructure as the tools needed to build a cloud. In order to host services and applications in the cloud, you need cloud infrastructure.

An abstraction technology or process—like virtualization—is used to separate resources from physical hardware and pool them into clouds; automation software and management tools allocate these resources and provision new environments so users can access what they need—when they need it. A cloud infrastructure as a service (IaaS), just like with all cloud technology, is accessed via the internet through a cloud vendor's data centre, which is responsible for maintaining and managing traditional on-premise hardware like servers and other storage devices as well as networking and visualization. That means the customer has the freedom and control to manage application, data, middleware, and other operating systems. The need for cloud infrastructure is to provide business scalability while consolidation. All of this activity is done by a cloud infrastructure management interface which is an open standard API requirement for handling cloud infrastructure. Use of cloud infrastructure efficiently helps in cost cutting. Without a cloud infrastructure a company has to add up all the hardware, software servers and energy bills. It also provides agility, and flexibility. Most of the cloud infrastructure services can be modified within seconds as compared to others services, which ultimately results in time saving and efficiency of business system. Cloud infrastructure aims to provide and improve security. Cloud infrastructure technologies and providers are always improving protection against hackers, viruses, and other data breaches with stronger firewalls, advanced encryption keys, and a hybrid approach that stores sensitive data in a private cloud and other data, even apps, in a public cloud.


Cloud infrastructure components
User applications and services
Application development and management tools
Cloud infrastructure management services
Networking
Storage
Servers and virtualization
Data center
©2017 TECHTARGET. ALL RIGHTS RESERVED TechTarget

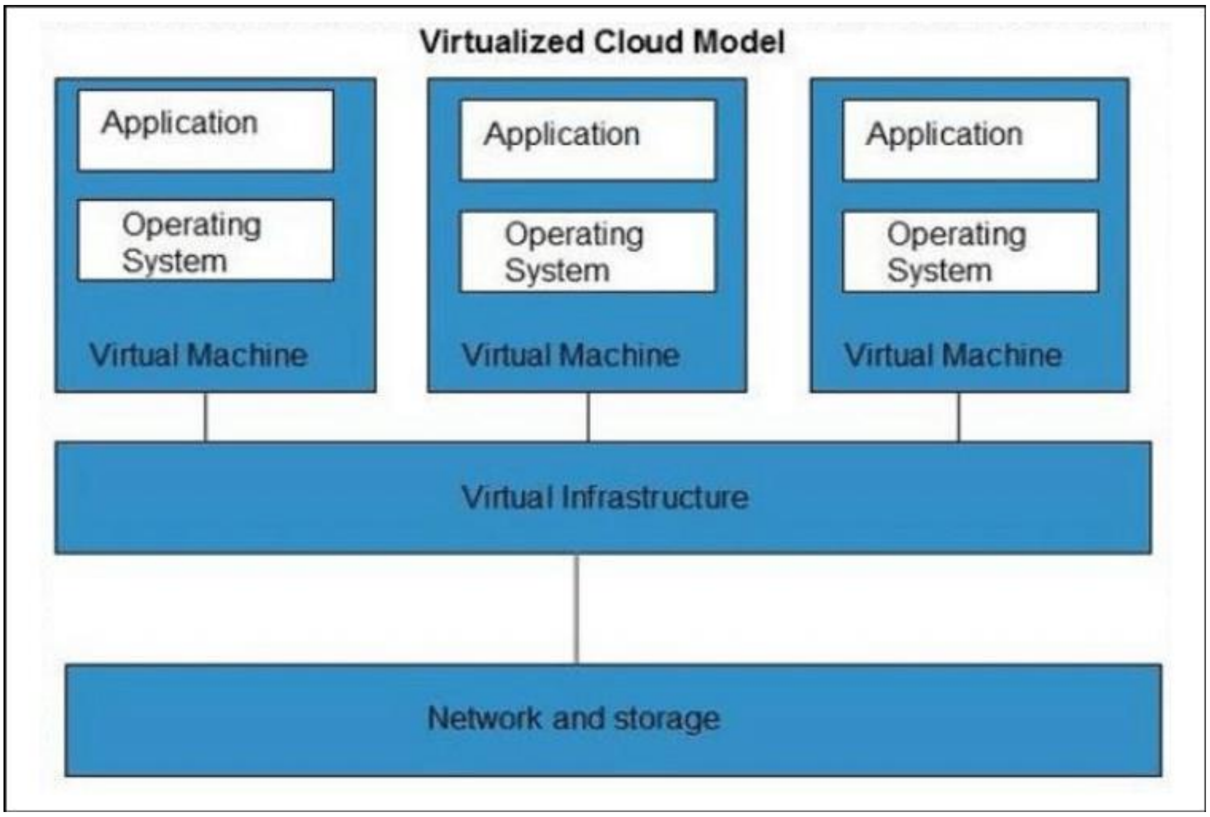
Virtualization-

Virtualization is a technique, which allows to share a single physical instance of a resource or an application among multiple customers and organizations. It does by assigning a logical name to a physical storage and providing a pointer to that physical resource when demanded. Most commonly, it refers to running multiple operating systems on a computer system simultaneously. To the applications running on top of the virtualized machine, it can appear as if they are on their own dedicated machine, where the operating system, libraries, and other programs are unique to the guest virtualized system and unconnected to the host operating system which sits below it. In the context of cloud computing, virtualization is a technique that makes a virtual ecosystem of storage devices and the server OS. Since cloud computing is being considered as a service or an application, assisting a virtualized ecosystem that could either be private or public, so with virtualization, resources could be escalated, reducing the necessity for a physical system.

Characteristics of virtualization:

- Resource distribution
- Isolation
- Availability
- Aggregation
- Authenticity and security

Virtualization provides high security through firewall which provides extra security from any sort of cyber threat and cyber-attacks. Virtualization follows many protocols which provide end to end security. It provides vast scope of flexibility. Users can work more efficiently as the work process is streamlined and agile. In virtualization, data can be transferred to virtual servers anytime and also be retrieved due to this users or cloud providers need not to waste time in finding out hard drives to discover data. It can also be described in such a way that with the help of Hypervisor which is software the cloud client can access the server. The hypervisor communicates between the server and the physical environment and distributes resources between different viewing environments.



Cloud task scheduling-

Cloud computing is an accelerating technology in the field of distributed computing. Cloud computing can be used in applications that include storing data, data analytics and IoT applications. Cloud computing is a technology that has changed traditional ways in which services are deployed by enterprises or individuals. It provides different types of services to registered users as web services so that the users do not need to invest in computing infrastructure. Task scheduling is the process of arranging incoming requests (tasks) in a certain manner so that the available resources will be properly utilized. Because cloud computing is the technology that delivers services through the medium of the Internet, service users must submit their requests online. While task planning techniques are used to determine the sequence in which tasks or tasks should be completed. It focuses on mapping user activities to available resources. The backbone of the work schedule is virtualization

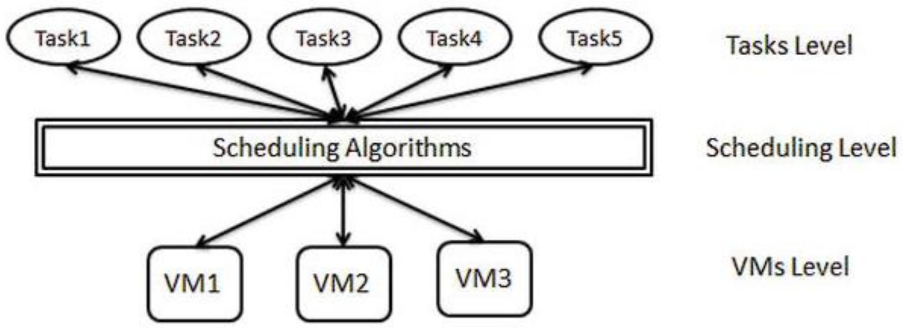
Let the task set be and the resource node collection is, in which. Task scheduling under cloud computing can be represented by the following matrix:

In the matrix, is 1, which means the task is performed on task, or is 0. This article defines the attributes of each resource node including processing capacity (that is, processing time), initial memory (that is, the load of response processing), and resource bandwidth (that is, reflecting the cost of processing). Therefore, the three vectors of virtual machine resources are the processing power vector, load capacity vector, and resource bandwidth vector. At the same time, each task corresponds to three matrices, which are the processing power vector, load capacity vector, and resource bandwidth vector; are the unit prices. Therefore, the time target function, load target function, and cost target function are shown as follows.

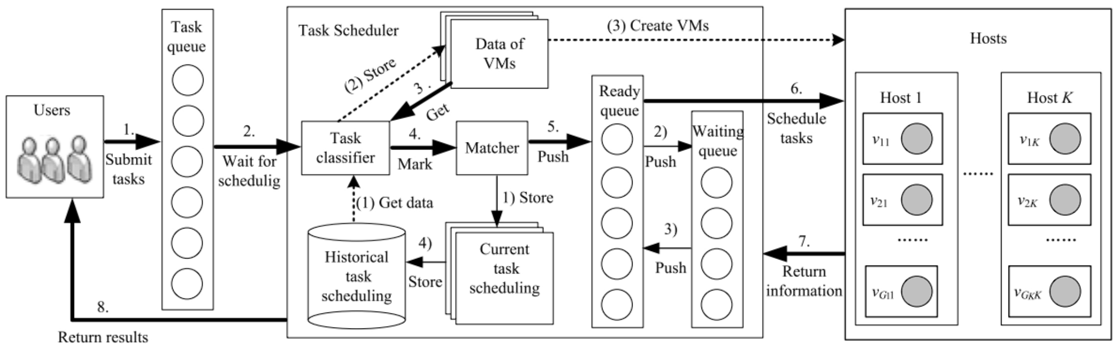
$$\text{time} = \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}\frac{E_{t,i}}{E_{n,j}},$$

$$\text{load} = \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}\frac{S_{t,i}}{S_{n,j}},$$

$$\text{cost} = \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}\frac{E_{t,i}}{E_{n,j}} \times \frac{C_{t,i}}{C_{n,j}} \times P.$$



\



<u>Framework of cloud task scheduling</u>

Problem Statement-

Cloud computing deals with different kinds of virtualized resources, hence scheduling places an important role in cloud computing. In cloud user may use hundreds of thousands of virtualized resources for each user task. So manual scheduling is not a feasible solution. Therefore, an optimized task scheduling algorithm is needed to find the most optimal solution.

LITERATURE REVIEW

Review of existing models-

S No.	**Name**	**Brief**	**Link**
1.	A Task Scheduling Algorithm with Improved Makespan Based on Prediction of Tasks Computation Time algorithm for Cloud Computing	This author has proposed a novel scheduling algorithm using Directed Acyclic Graph (DAG) based on the Prediction of Tasks Computation Time algorithm (PTCT) to estimate the preeminent scheduling algorithm for prominent cloud data using Min-Min, Max-Min, QoS-Guide and MiM-MaM scheduling algorithms. Used parameters are Tasks([10-90]), Machines(8), Algorithms(4), Type of data(High, Medium and low QoS). This permits reduction of the size of Expected Time to Compute(ETC) matrix.	https://sci-hub.se/10.1109/ACCESS.2019.2948704
2.	W-Scheduler: whale optimization for task scheduling in cloud computing	This author has proposed a task scheduling algorithm for scheduling the tasks to the virtual machines in the cloud computing environments based on the multi-objective model and the whale optimization algorithm. Used parameters: memory requirement, cost, deadline, and the required budget. Proposed W-Scheduler can optimally schedule the tasks to the virtual machines while consuming minimum make span of 7 and minimum average cost of 5.8.	https://sci-hub.ee/10.1007/s10586-017-1055-5
3.	A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems	This author has proposed a WOA(Whale optimization algorithm)-based task scheduling approach for cloud computing, which aims to improve the performance of a system with given computing resources. Used parameters are ACO(pheromone evaporation coefficient, path selection probability, inertia weight), PSO(acceleration constant), IWC(initial population, largest population, nonlinear factor, individual generate parameter). . It can greatly improve the efficiency of a cloud computing system, in terms of both system load and system resource utilization cost.	https://sci-hub.ee/10.1109/JSYST.2019.2960088
4.	A PSO Algorithm Based Task Scheduling in Cloud Computing	This author has proposed a PSO algorithm-based task scheduling in order to reduce the overall make span along with the increase in the utilization of underlying computing resources. Used parameters	https://sci-hub.ee/10.4018/ijamc.20191001

		population size (25), Maximum iterations (100), Crossover type and probability, mutual probability, mutation operator, selection operator. the proposed strategy reduces make span by around 30% while increasing the resource utilization by 20% approximately and outperforms its peers which are taken into consideration.	01
5.	Task Scheduling in Cloud Computing Environment by Grey Wolf Optimizer	This author has proposed GWO swarm intelligence metaheuristics for tackling task scheduling problems in cloud computing environments. Performed improvements of the GWO over other heuristics and metaheuristics expressed in percentages. Used parameters are Number of tasks, Makespan value, iterations. Results clearly indicate that the GWO has potential in tackling this kind of problem.	https://sci-hub.ee/10.1109/TELFOR48224.2019.8971223
6.	Task Scheduling in Cloud Computing Based on Hybrid Moth Search Algorithm and Differential Evolution	This author has proposed MSDE algorithm as an alternative method to solve the task scheduling problem in cloud computing environment, this algorithm combining the moth search algorithm (MSA) and with the differential evolution (DE). Used parameters are No. of hosts, No. of cloudlets, length etc. The main objective is to enhance the throughput of the cloud system by minimizing the makespan.	https://sci-hub.ee/10.1016/j.knosys.2019.01.023
7.	Chaotic social spider algorithm for load balance aware task scheduling in cloud computing	This author has proposed CSSA for load balance aware task scheduling in cloud computing environments. This algorithm was inspired by the foraging behavior of social spider species and its relationships. Used parameters are No. of VM, No. of tasks, Bandwidth, Cost per VM, MIPS, RAM, No. of physical machines. The main objective was minimizing the makespan of the cloud scheduling process thereby improving the throughput of the cloud system	https://sci-hub.ee/10.1007/s10586-018-1823-x
8.	Amelioration of task scheduling in cloud computing using crow search algorithm	This author has proposed nature-inspired CSA emphasizes on the makespan reduction in task scheduling. Based on the 12 cases of task and heterogeneity of VM, experimental tests are conducted and makespan values are calculated for CSA, Min–Min and ACO algorithms. Used parameter no. of task, no. of VM, Awareness probability etc. Results show that CSA reduces the	https://sci-hub.ee/10.1007/s00521-019-04067-2

		makespan value compared to ACO and Min–Min algorithms.	
9.	An EDA-GA Hybrid Algorithm for Multi-Objective Task Scheduling in Cloud Computing	This author has proposed an effective EDA-GA hybrid algorithm to address the multi-objective task scheduling problem with the goal of reducing the task completion time and improving the system load balancing ability. Used parameters are PS (size of the population), E (size of the elite population, E = PS × p%), and λ (learning rate). The results show that the proposed EDA-GA hybrid algorithm has good convergence speed and search ability, and it performs better in reducing task completion time and improving load balancing ability.	https://sci-hub.ee/10.1109/access.2019.2946216
10.	An Enhanced Task Scheduling in Cloud Computing Based on Hybrid Approach	This author has proposed a hybrid shortest–longest scheduling algorithm which tries to handle the starvation problem, which is a very sensitive issue, where most of the state-of-the-art scheduling algorithms suffer. Used parameters are Number of data centre, Number of cloud hosts, Host MIPS, Number of CPUs per host, Virtual machine bandwidth etc. The experimental results proved that the proposed HSLJF algorithm has outperformed the existing algorithms in minimizing the average make span, response time, and the actual execution time of tasks while maximizing resource utilization and throughput.	https://sci-hub.ee/10.1007/978-981-13-2514-4_2

PROPOSED MODEL

Proposed solution-

Finding an optimal solution for cloud task scheduling using a machine learning algorithm. Comparing with another machine learning algorithm. Work on parameter like Total execution time, fault rate, number of tasks failed, number of tasks completed, start time, finish time. Conclude that the algorithm is Good or Bad

Tools Required-

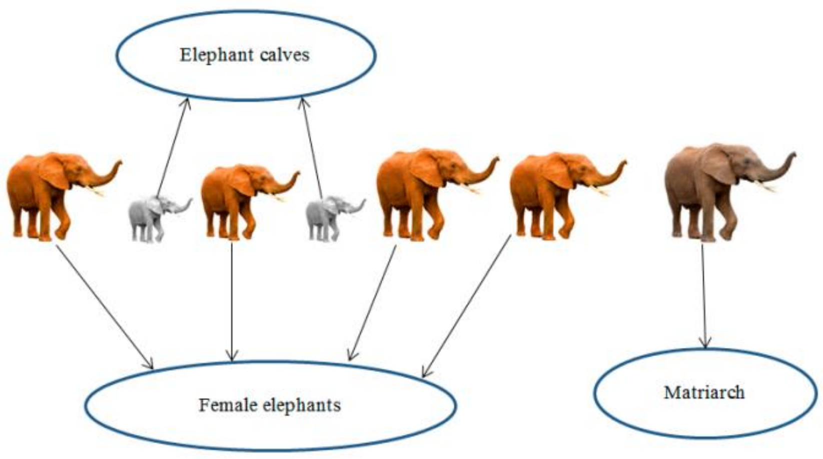
- Cloud-Sim 3.0
- NetBeans
- Python

PROPOSED ALGORTIHM-

Abstract: Elephant herding optimization (EHO) is a nature-inspired metaheuristic optimization algorithm based on the herding behaviour of elephants. EHO uses a clan operator to update the distance of the elephants in each clan with respect to the position of a matriarch elephant. The superiority of the EHO method to several state-of-the-art metaheuristic algorithms has been demonstrated for many benchmark problems and in various application areas.

Basics of Elephant Herding Optimization-

Elephants, as social creatures, live in social structures of females and calves. An elephant clan is headed by a matriarch and composed of a number of elephants. Female members like to live with family members, while the male members tend to live elsewhere. le members like to live with family members, while the male members tend to live elsewhere. They will gradually become independent of their families until they leave their families completely. The EHO technique proposed by Wang et al. in 2015 was developed by after studying natural elephant herding behavior.



1. Some clans with fixed numbers of elephants comprise the elephant population.
2. A fixed number of male elephants will leave their family group and live solitarily far away from the main elephant group in each generation.
3. A matriarch leads the elephant in each clan.

Clan-updating Operator-

According to the natural habits of elephants, a matriarch leads the elephants in each clan. Therefore, the new position of each elephant *ci* is influenced by matriarch *ci*. The elephant *j* in clan *ci* can be calculated by Equation (1).

$$x_{new,ci,j} = x_{ci,j} + a \times (x_{best,ci} - x_{ci,j}) \times r \tag{1}$$

where $x_{new,ci,j}$ and $x_{ci,j}$ present the new position and old position for elephant *j* in clan *ci*, respectively. $x_{best,ci}$ is matriarch c_i which represents the best elephant in the clan. $a \in [0,1]$ indicates a scale factor, $r \in [0,1]$. The best elephant can be calculated by Equation (2) for each clan.

$$x_{new,ci,j} = \beta \times x_{center,ci} \tag{2}$$

where $\beta \in [0,1]$ represents a factor which determines the influence of the $x_{center,ci}$ on $x_{new,ci,j}$. $x_{new,ci,j}$ is the new individual. $x_{center,ci}$ is the center individual of clan *ci*. It can be calculated by Equation (3) for the *d*-th dimension.

$$x_{center,ci,d} = \frac{1}{n_{ci}} \times \sum_{j=1}^{n_{ci}} x_{ci,j,d}$$

where $1 \leq d \leq D$ and n_{ci} indicate the number of elephants in clan *ci*. $x_{ci,j,d}$ represents the *d*-th dimension

of elephant individual $x_{ci,j}$. $x_{center,ci}$ is the center of clan *ci* and it can be updated by Equation (3).

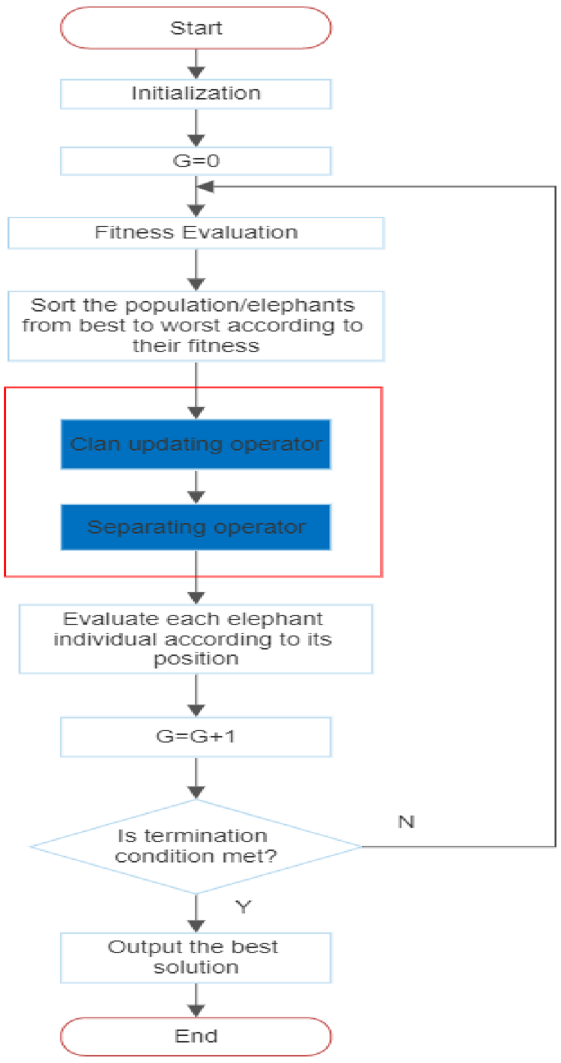
Separating Operator-

The separating process whereby male elephants leave their family group can be modeled into separating operator when solving optimization problems. The separating operator is implemented by the elephant individual with the worst fitness in each generation, as shown in Equation (4).

$$x_{worst,ci} = x_{\min} + (x_{\max} - x_{\min} + 1) \times rand \tag{4}$$

where $x_{\max}$ represents the upper bound of the individual and $x_{\min}$ indicates lower bound of the individual. $x_{worst,ci}$ indicates the worst individual in clan c_i. *Rand* [0, 1] is a stochastic distribution between 0 and 1.

According to the description of the clan-updating operator and separating operator, the mainframe of EHO is summarized. The corresponding flowchart is shown as follows. *MaxGen* is the maximum generation.


Start
Initialization
G=0
Fitness Evaluation
Sort the population/elephants from best to worst according to their fitness
Clan updating operator
Separating operator
Evaluate each elephant individual according to its position
G=G+1
Is termination condition met?
N
Y
Output the best solution
End

<u>**Flowchart of Elephant Herd Optimization**</u>

Algorithm-

```
1.  Begin
2.  Initialization. Set the initialize iterations $G = 1$; initialize the population $P$ randomly; set
    maximum generation MaxGen
3.  While stopping criterion is not met do
4.      Sort the population according to fitness of individuals.
5.      For all clans $ci$ do
6.        For elephant $j$ in the clan $ci$ do
7.          Generate $x_{new, ci,j}$ and update $x_{ci,j}$ by Equation (1).
8.         If $xci,j = xbest,ci$ then
9.           Generate $x_{new, ci,j}$ and update $x_{ci,j}$ by Equation (2).
10.         End if
11.        End for
12.       End for
13.         For all clans $ci$ do
14.           Replace the worst individual $ci$ by Equation (4).
15.         End for
16.         Evaluate each elephant individual according to its position.
17.         $T = T + 1$.
18.  End while
19. End.
```

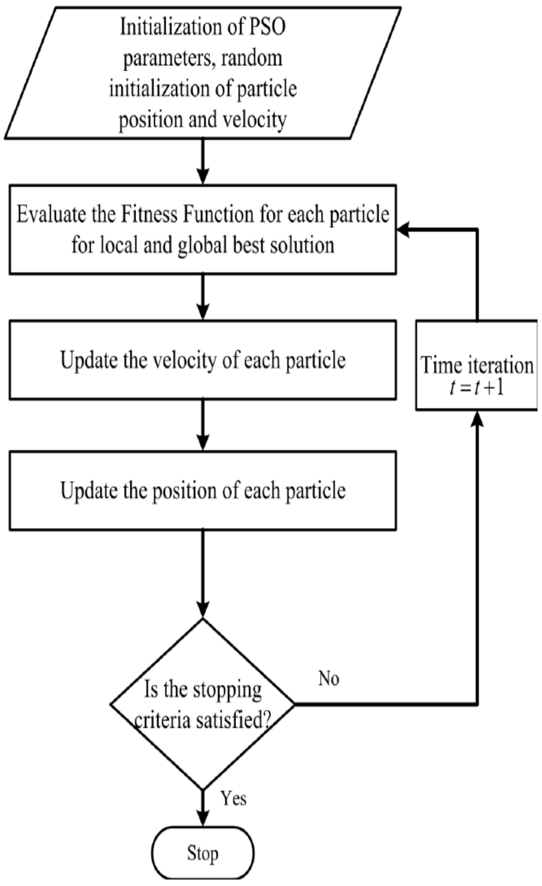
Particle Swarm Optimization (Algorithm used for comparison)-

Suppose there is a swarm (a group of birds). Now, all the birds are hungry and are searching for food. These hungry birds can be correlated with the tasks in a computation system which are hungry for resources.

Now, in the locality of these birds, there is only one food particle. This food particle can be correlated with a resource. **As we know, tasks are many, resources are limited.** So, this has become a similar condition as in a certain computation environment.

The birds don't know where the food particle is hidden or located. In such a scenario, how the algorithm to find the food particle should be designed**. If every bird will try to find the food on its own, it may cause havoc and may consume a large amount of time.** Thus, on careful observation of this swarm, it was realized that though the birds don't know where the food particle is located, they do know their distance from it.

Thus, the best approach to finding that food particle is **to follow the birds which are nearest to the food particle**. This behavior of birds is simulated in the computation environment and the algorithm so designed is termed as **Particle Swarm Optimization Algorithm.**


Initialization of PSO parameters, random initialization of particle position and velocity
Evaluate the Fitness Function for each particle for local and global best solution
Update the velocity of each particle
Update the position of each particle
Is the stopping criteria satisfied?
No
Yes
Stop
Time iteration $t=t+1$

Flow Chart for Particle Swarm Optimization

In PSO, each candidate solution is called a "particle" and represents a point in a D-dimensional space, if D is the number of parameters to be optimized. Accordingly, the position of the i th particle may be described by the vector xi:

$$\mathbf{x}_i = \left[x_{i1}x_{i2}x_{i3}\cdots x_{iD}\right]$$

and the population of N candidate solutions constitutes the swarm:

$$\mathbf{X} = \left\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\right\}$$

In searching for the optimal solution of the problem, the particles define trajectories in the parameter space (i.e., iteratively update their positions) based on the following equation of motion:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$$

the velocity of the i th particle is defined as:

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1\left(\mathbf{p}_i - \mathbf{x}_i(t)\right)\mathbf{R}_1 + c_2\left(\mathbf{g} - \mathbf{x}_i(t)\right)\mathbf{R}_2$$

Python Module Used-

MEALPY is a Python library for the most of cutting-edge population *meta-heuristic algorithms* - a field which provides a fast and efficient way to find the global optimal point of mathematical optimization problems.

Population meta-heuristic algorithms (PMA) are the most popular algorithm in the field of optimization. There are several types of PMA such as:

- Evolutionary inspired computing
- Swarm inspired computing
- Physics inspired computing
- Human inspired computing
- Biology inspired computing
- And others such as: Music inspired, Mathematical inspired computing.

Fitness Function-

```
vm=10
tcount=10000
number=1500
f = open('data_'+str(tcount)+"_"+str(vm)+'.json')
data = json.load(f)
size=number
vmcount= data['vmcount'][0]

problem_dict1 = {
   "fit_func": Fun,
   "lb": [0 for i in range(size)] ,
   "ub": [vmcount for i in range(size)],
   "minmax": "min",
   "verbose": True,
}
model1 = BasePSO(problem_dict1, epoch=100, pop_size=100, pr=0.03)
model1.solve()
```

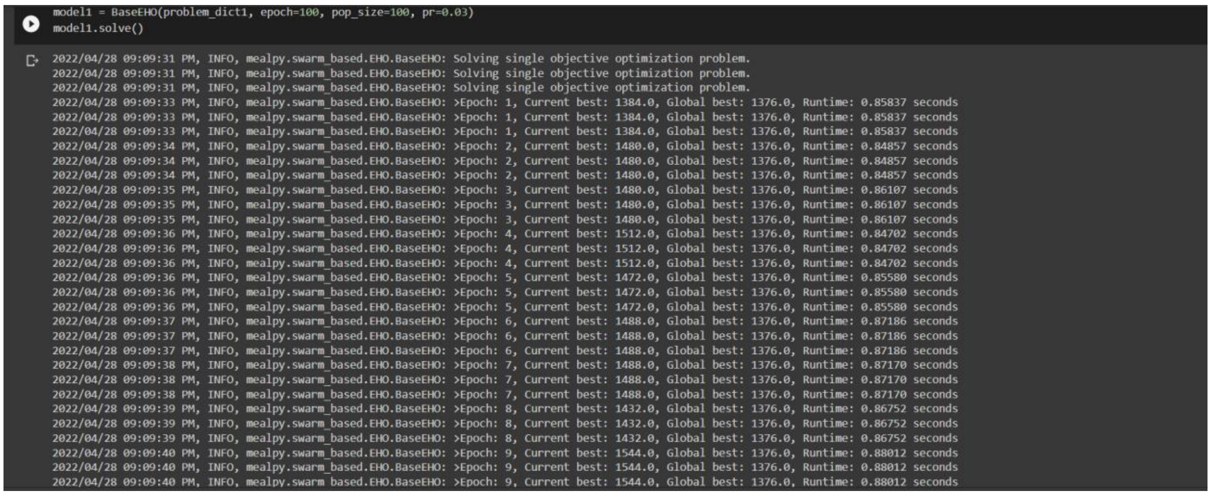
CloudSim-

It is a simulation toolkit that supports the modeling and simulation of the core functionality of cloud, like job/task queue, processing of events, creation of cloud entities (datacenter, datacenter brokers, etc.), communication between different entities, implementation of broker policies, etc. This toolkit allows to:

- Test application services in a repeatable and controllable environment.
- Tune the system bottlenecks before deploying apps in an actual cloud.
- Experiment with different workload mix and resource performance scenarios on simulated infrastructure for developing and testing adaptive application provisioning techniques.

RESULTS

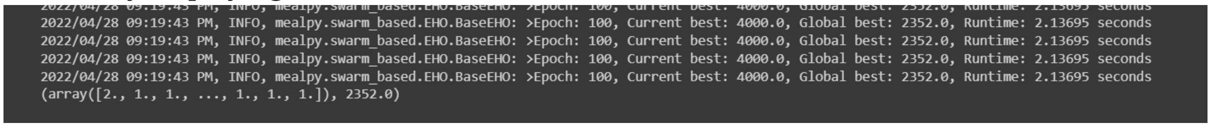
Code Output-



```
model1 = BaseEHO(problem_dict1, epoch=100, pop_size=100, pr=0.03)
model1.solve()

2022/04/28 09:09:31 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: Solving single objective optimization problem.
2022/04/28 09:09:31 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: Solving single objective optimization problem.
2022/04/28 09:09:31 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: Solving single objective optimization problem.
2022/04/28 09:09:33 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 1, Current best: 1384.0, Global best: 1376.0, Runtime: 0.85837 seconds
2022/04/28 09:09:33 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 1, Current best: 1384.0, Global best: 1376.0, Runtime: 0.85837 seconds
2022/04/28 09:09:33 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 1, Current best: 1384.0, Global best: 1376.0, Runtime: 0.85837 seconds
2022/04/28 09:09:34 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 2, Current best: 1480.0, Global best: 1376.0, Runtime: 0.84857 seconds
2022/04/28 09:09:34 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 2, Current best: 1480.0, Global best: 1376.0, Runtime: 0.84857 seconds
2022/04/28 09:09:34 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 2, Current best: 1480.0, Global best: 1376.0, Runtime: 0.84857 seconds
2022/04/28 09:09:35 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 3, Current best: 1480.0, Global best: 1376.0, Runtime: 0.86107 seconds
2022/04/28 09:09:35 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 3, Current best: 1480.0, Global best: 1376.0, Runtime: 0.86107 seconds
2022/04/28 09:09:35 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 3, Current best: 1480.0, Global best: 1376.0, Runtime: 0.86107 seconds
2022/04/28 09:09:36 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 4, Current best: 1512.0, Global best: 1376.0, Runtime: 0.84702 seconds
2022/04/28 09:09:36 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 4, Current best: 1512.0, Global best: 1376.0, Runtime: 0.84702 seconds
2022/04/28 09:09:36 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 4, Current best: 1512.0, Global best: 1376.0, Runtime: 0.84702 seconds
2022/04/28 09:09:36 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 5, Current best: 1472.0, Global best: 1376.0, Runtime: 0.85580 seconds
2022/04/28 09:09:36 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 5, Current best: 1472.0, Global best: 1376.0, Runtime: 0.85580 seconds
2022/04/28 09:09:36 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 5, Current best: 1472.0, Global best: 1376.0, Runtime: 0.85580 seconds
2022/04/28 09:09:37 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 6, Current best: 1488.0, Global best: 1376.0, Runtime: 0.87186 seconds
2022/04/28 09:09:37 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 6, Current best: 1488.0, Global best: 1376.0, Runtime: 0.87186 seconds
2022/04/28 09:09:37 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 6, Current best: 1488.0, Global best: 1376.0, Runtime: 0.87186 seconds
2022/04/28 09:09:38 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 7, Current best: 1488.0, Global best: 1376.0, Runtime: 0.87170 seconds
2022/04/28 09:09:38 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 7, Current best: 1488.0, Global best: 1376.0, Runtime: 0.87170 seconds
2022/04/28 09:09:38 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 7, Current best: 1488.0, Global best: 1376.0, Runtime: 0.87170 seconds
2022/04/28 09:09:39 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 8, Current best: 1432.0, Global best: 1376.0, Runtime: 0.86752 seconds
2022/04/28 09:09:39 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 8, Current best: 1432.0, Global best: 1376.0, Runtime: 0.86752 seconds
2022/04/28 09:09:39 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 8, Current best: 1432.0, Global best: 1376.0, Runtime: 0.86752 seconds
2022/04/28 09:09:40 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 9, Current best: 1544.0, Global best: 1376.0, Runtime: 0.88012 seconds
2022/04/28 09:09:40 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 9, Current best: 1544.0, Global best: 1376.0, Runtime: 0.88012 seconds
2022/04/28 09:09:40 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 9, Current best: 1544.0, Global best: 1376.0, Runtime: 0.88012 seconds
```

Final array displaying Global Best.



```
2022/04/28 09:19:43 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 100, Current best: 4000.0, Global best: 2352.0, Runtime: 2.13695 seconds
2022/04/28 09:19:43 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 100, Current best: 4000.0, Global best: 2352.0, Runtime: 2.13695 seconds
2022/04/28 09:19:43 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 100, Current best: 4000.0, Global best: 2352.0, Runtime: 2.13695 seconds
2022/04/28 09:19:43 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 100, Current best: 4000.0, Global best: 2352.0, Runtime: 2.13695 seconds
2022/04/28 09:19:43 PM, INFO, mealpy.swarm_based.EHO.BaseEHO: >Epoch: 100, Current best: 4000.0, Global best: 2352.0, Runtime: 2.13695 seconds
(array([2., 1., 1., ..., 1., 1., 1.]), 2352.0)
```

Comparing Execution Time (Global Best) of Elephant herd optimization VS Particle swarm optimization-

Parameters Used-

- No. of VMs (5,10)
- Epoch
- Population Size
- No. of task
- Global best

When VM=5

Epoch	Population Size	No. of Task	Global Best (EHO)	Global Best (PSO)
100	100	100	112	95.9
100	100	500	192	571
100	100	1000	824	1187
100	100	1500	992	1733
100	100	2000	2272	2318
100	100	2500	2352	2816
100	100	3000	368	3550

When VM=10

Epoch	Population Size	No. of Task	Global Best (EHO)	Global Best (PSO)
100	100	100	62.8	64
100	100	500	336	300
100	100	1000	639.9	611
100	100	1500	952	902
100	100	2000	1280	1228
100	100	2500	1664	1540
100	100	3000	2040	1714

Findings-

After comparing the dataset by testing it with increasing no. of tasks and by keeping other parameters same we observed that when the resources were less like in VM=5 table EHO gave better results compared to PSO but when tasks were very less(appx.100) PSO was better.
But when the resources were more like in VM=10 table we didn't saw much difference in our results so when resources are more both the algorithms gave good results.
Comparing both the tables we can say using EHO is good.

Comparing Execution Time (Global Best) of Elephant herd optimization VS Particle swarm optimization-

Parameters Used-

- No. of VMs (5,10)
- Epoch
- Population Size
- No. of task
- Global best

When VM=5

Epoch	Population Size	No. of Task	Global Best (EHO)	Global Best (PSO)
100	100	500	192	562
100	200	500	628	524
100	300	500	616	524.57
100	100	1500	992	1838.85
100	200	1500	2040	1747
100	300	1500	2088	1588

When VM=10

Epoch	Population Size	No. of Task	Global Best (EHO)	Global Best (PSO)
100	100	500	336	305.14
100	200	500	308	301
100	300	500	314	297
100	100	1500	952	908.57
100	200	1500	976	920
100	300	1500	937	913

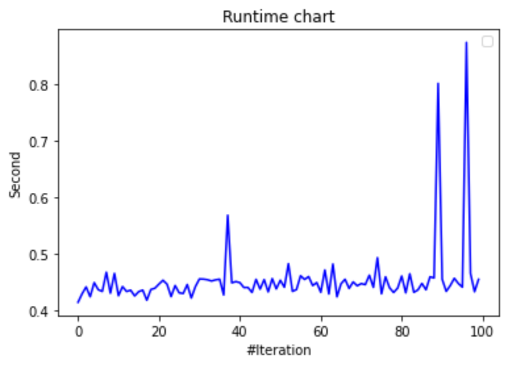
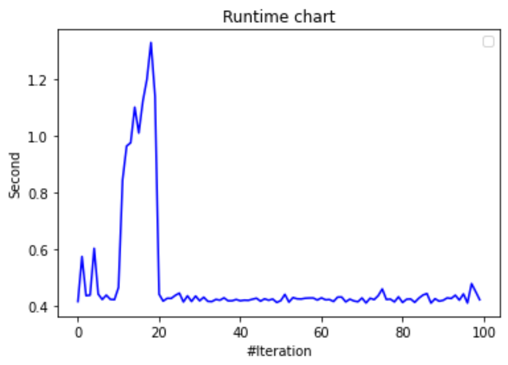
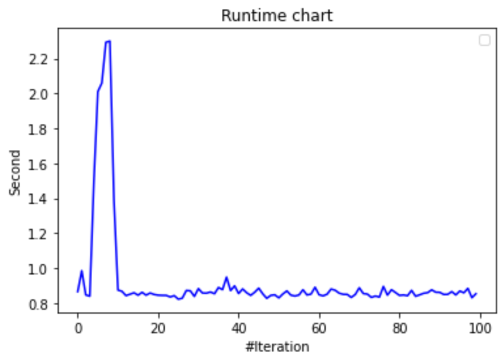
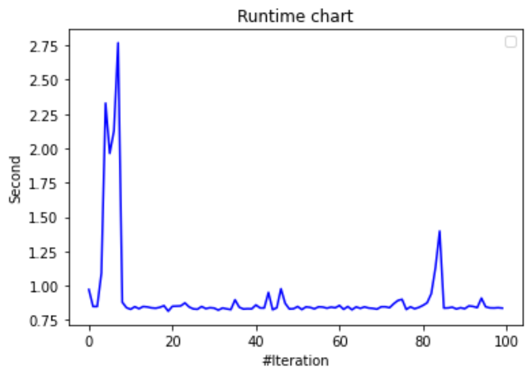
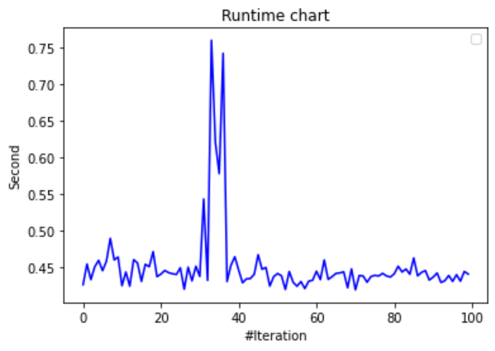
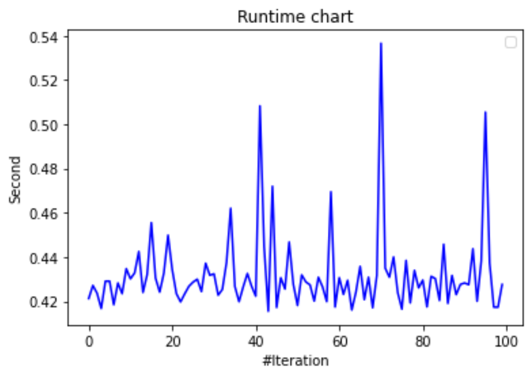
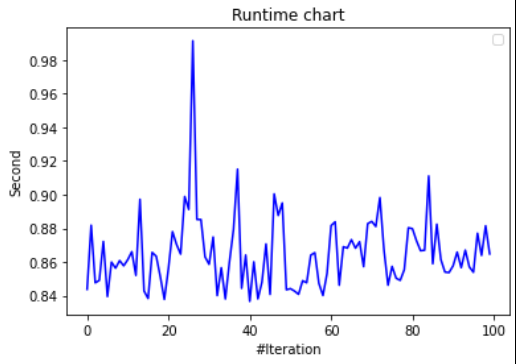
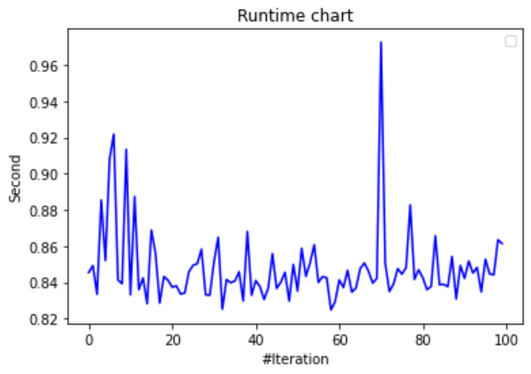
Findings-

When we increase the population size in both the algorithm we didn’t saw any major change in the global best. Only when population size was less in 5 VM we saw some differences in EHO.

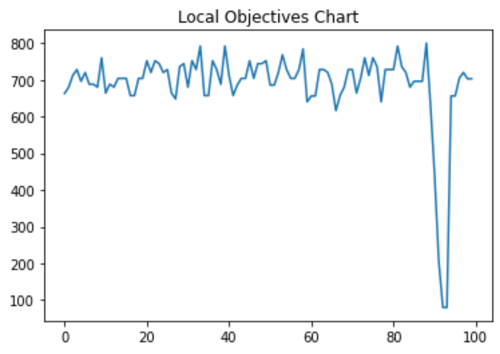
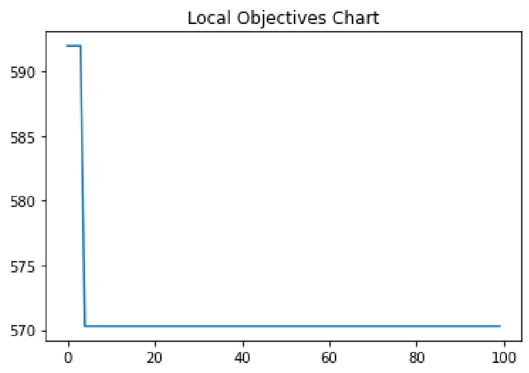
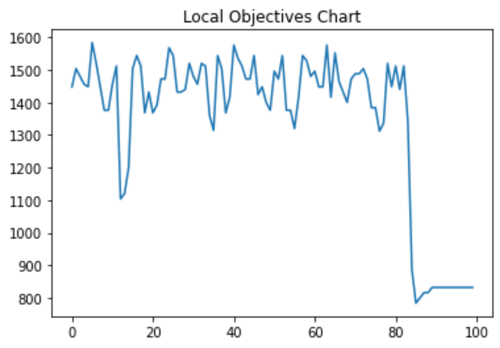
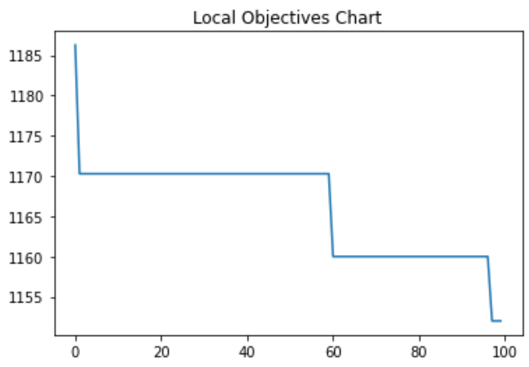
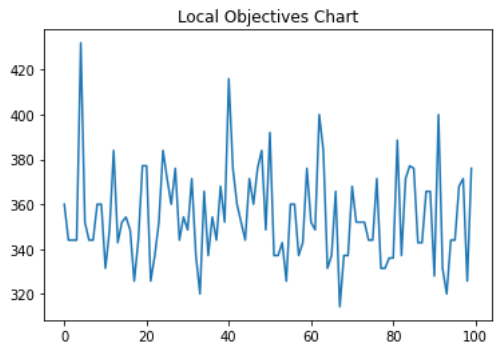
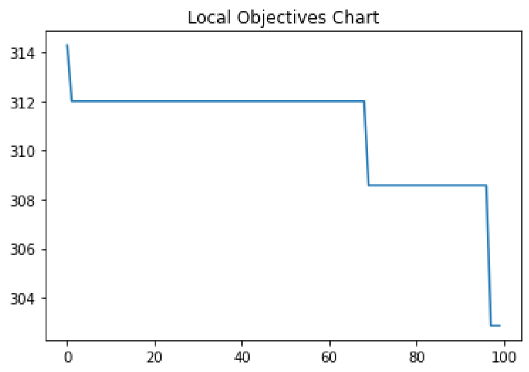
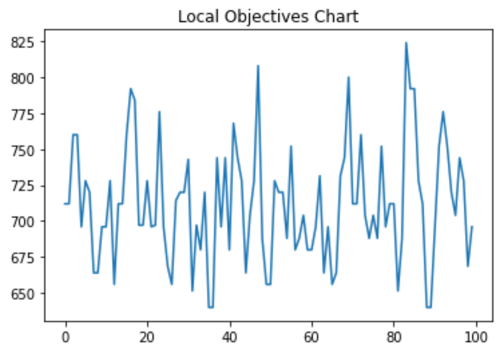
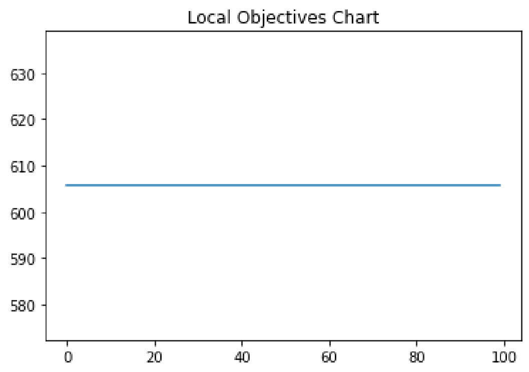
Comparing Elephant herd optimization VS Particle swarm optimization by Visualizing–

S No.	Epoch	Population Size	No. of Task	No. of VM	Global Best (EHO)	Global Best (PSO)
1	100	100	500	5	80	541
2	100	200	1000	5	784	1147
3	100	300	500	10	314.28	297
4	100	100	1000	10	639	634

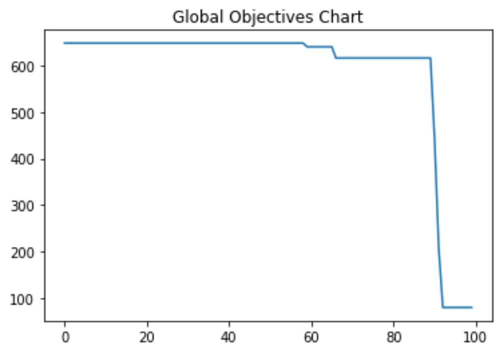
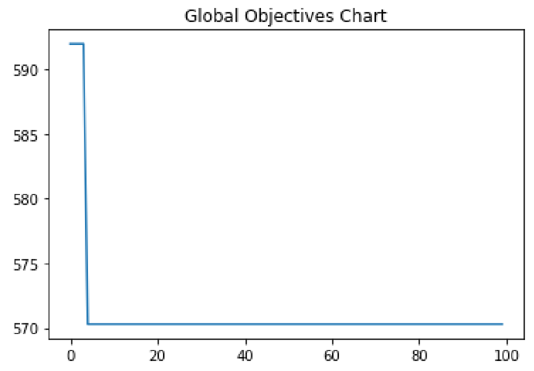
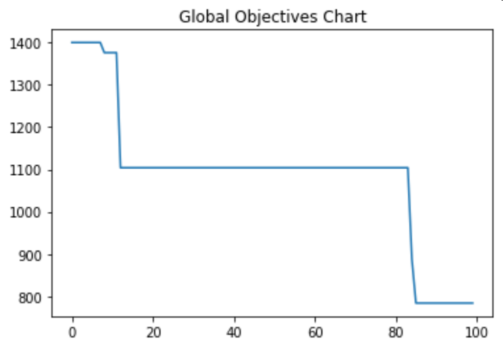
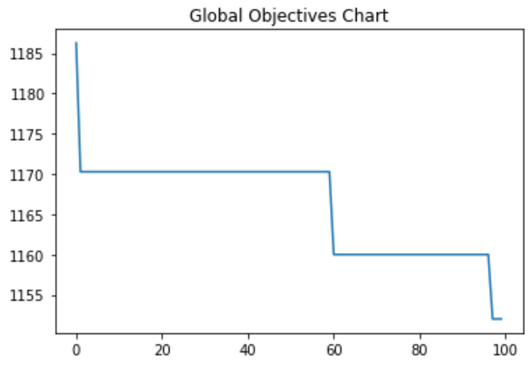
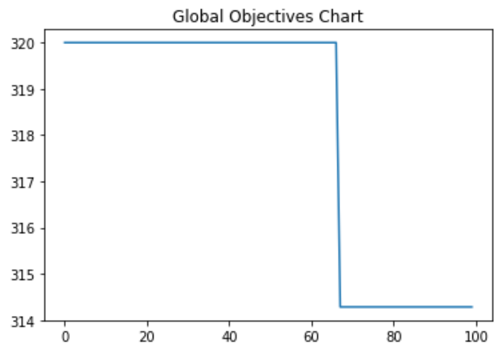
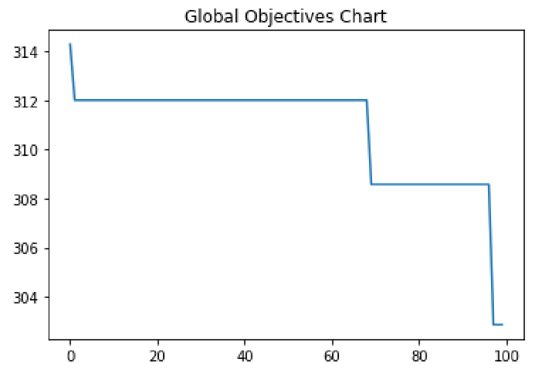
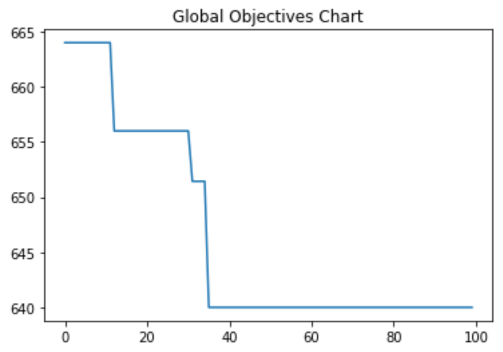
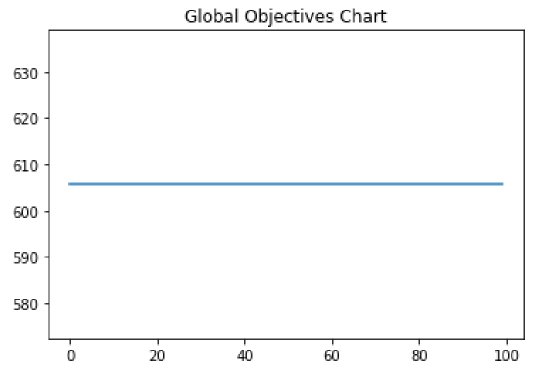
1. Comparing Runtime Per Epoch-

S No.	EHO	PSO
1		
2		
3		
4		

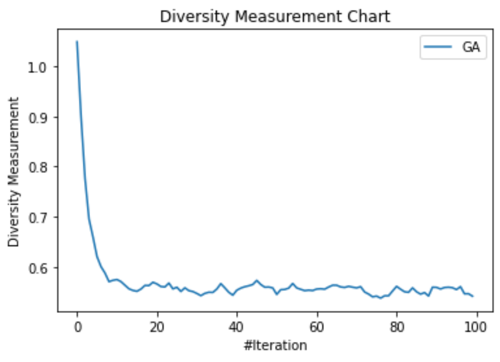
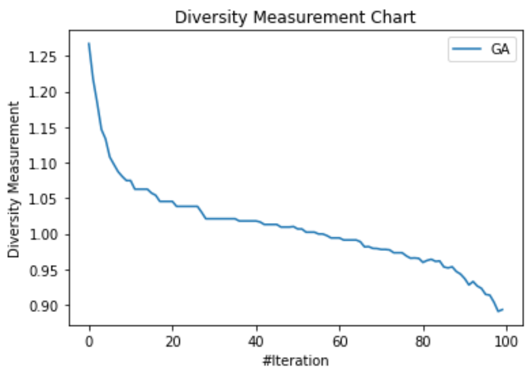
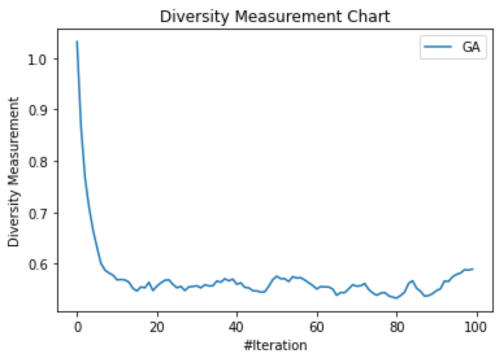
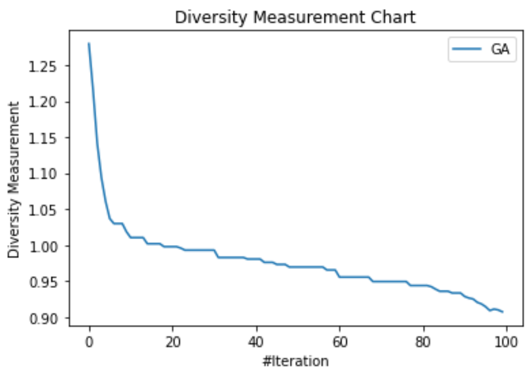
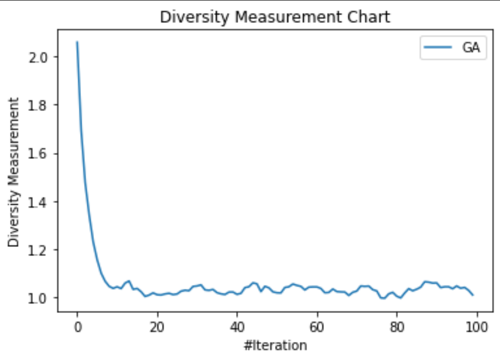
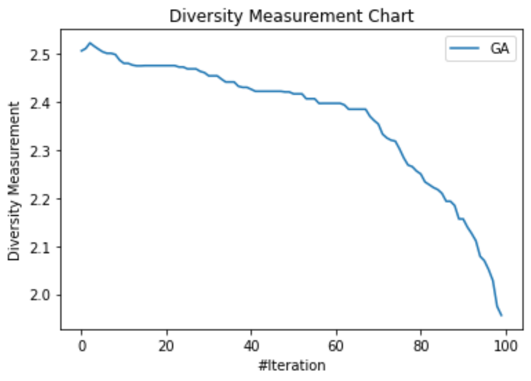
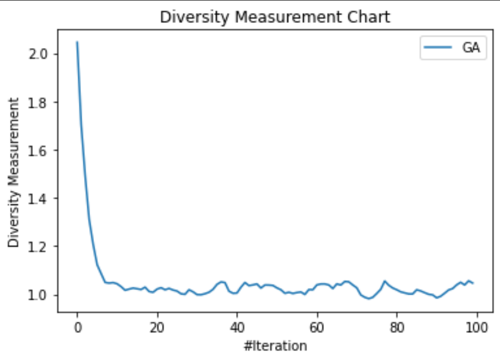
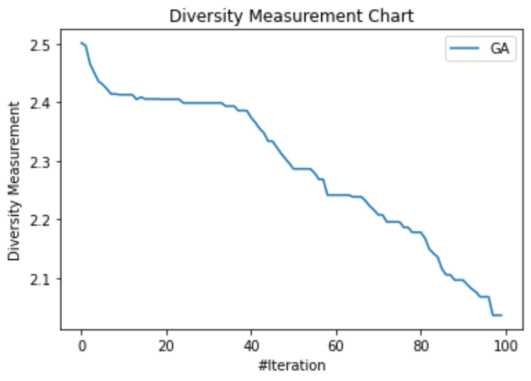
2. Comparing Local Best by Local Objective chart-

S No.	EHO	PSO
1		
2		
3		
4		

3. Comparing Global Best by Global Objective chart-

S No.	EHO	PSO
1		
2		
3		
4		

4. Comparing Diversity Measurement chart-

S No.	EHO	PSO
1		
2		
3		
4		

CONCLUSION

Task Scheduling in cloud is one of the most challenging things. So, we have proposed the Elephant Herd Optimization (EHO) as a method to solve cloud task scheduling problems in cloud. The EHO algorithm is inspired by behavior of elephant herd and its relationship.

In our project we had done various performance analysis of our proposed algorithm EHO and compared it with another nature inspired machine learning algorithm Particle Swarm Optimization (PSO). At first, we had solved the set of tasks ranging from 100 task to 3000 tasks. Then we had solved the by keeping number of tasks constant and varying population size. At last, we had done both of our first task with higher number of virtual machines. The performance of the proposed algorithm to solve the task scheduling problem was compared with other heuristic algorithms for data. It can be concluded that the proposed algorithm had provided better performance than the existing algorithm in many scenarios.

We conclude that EHO has a good characteristic as optimization algorithm and it can be used for solving complex optimizations problems.

In future we will be working on simulating this entire thing on CloudSim simulation environment and will enhance the throughput of cloud system by minimizing the execution time of the assigned with the minimum wastage of cloud resource from the virtual machines.

REFERENCES

- https://link.springer.com/article/10.1007/s00500-016-2474-6
- https://www.hindawi.com/journals/wcmc/2021/4888154/
- https://www.researchgate.net/publication/350801467_Task_Scheduling_Algorithms_in_Cloud_Computing_A_Review
- https://ieeexplore.ieee.org/abstract/document/5476775
- https://www.geeksforgeeks.org/what-is-cloudsim/
- https://www.redhat.com/en/topics/cloud-computing/what-is-cloud-infrastructure
- https://www.cloudflare.com/en-in/learning/cloud/what-is-the-cloud/
- https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-018-0105-8