Particle Swarm Ant Colony Optimization Task Scheduling Algorithm in Cloud Computing Environment

By: Tefera Yehualashet

A Thesis Submitted to

Department of Computing

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Software Engineering

Office of Graduate Studies

Adama Science and Technology University

Adama

January, 2018

Particle Swarm Ant Colony Optimization Task Scheduling Algorithm in Cloud Computing Environment

By: Tefera Yehualashet

Advisor: Dr. Ravindra Babu

A Thesis Submitted to

Department of Computing

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Software Engineering

Office of Graduate Studies

Adama Science and Technology University

Adama

January, 2018

Declaration

I hereby declare that this MSc Thesis is my original work and has not been submitted somewhere else for a partial fulfillment of the requirement for the degree or any other qualification. All materials and related works used for this study has been mentioned and duly acknowledged in my own work.

Tefera Yehualashet Signature: _____

This thesis has been submitted for examination with my approval as university advisor.

Dr. Ravindra Babu Signature: _____

Date of submission: 26/01/2018

Adama

January, 2018

Approval of Board of Examiners

We, the undersigned, members of the Board of Examiners of the final open defense by Mr. **Tefera Yehualashet** have read and evaluated his/her thesis entitled "**A particle swarm ant colony optimization task scheduling algorithm in cloud computing environment**" and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirement of the Degree of Master of Science in Software Engineering.

Supervisor /Advisor	Signature	Date
Chairperson	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Acknowledgement

First of all, I would like to thanks to the almighty God for he gives me this glories chance and constant help. Next, I would like to thanks my respected advisor, Dr. Ravindra Babu, Department of Computing and Engineering. For his support and continuous help throughout the period this work was carried out. Finally, I would like to thank my friends and family they encouraged and assisted me throughout the completion of this thesis.

Acronyms and Abbreviations

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AWS	Amazon Web Services
DC	Data Center
Equ.	Equation
FCFS	First Come First Served
GA	Genetic Algorithm
GB	Giga Byte
Gbest	Global best
GUI	Graphical User Interface
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
IT	Information Technology
MB	Megabyte
MIPS	Million Instruction Per Second
NASDAQ	National Association of Securities Dealers Automated Quotations
MS	Millisecond
NIST	National Institute of Standards and Technology
PaaS	Platform as a Service
Pbest	Local best
PSACO	Particle Swarm Ant Colony Optimization
PSOCS	Particle Swarm Optimization Cuckoo Search
PSO	Particle Swarm Optimization
QoS	Quality of Service
RR	Round Robin
SaaS	Software as a Service
TSGA	Task scheduling Genetic Algorithm
UB	User Base
VM	Virtual Machine
WWW	World Wide Web

Table of Contents

List of Tables

List of Figures

Abstract

Cloud computing refers to a collaborative Information Technology environment, which is deliberate with the aim of measurable and remotely spreading scalable IT resources for effective and green utilization. Task scheduling in cloud environment is responsible to assign user tasks effectively to appropriate resources for execution. Task scheduling problem has always been an important issue in cloud environment. In order to improve the task scheduling issue in cloud environment the researcher community proposed different task scheduling algorithm.

The existing algorithms are trying to solve task scheduling problem based on different performance metrics but there are some problems are existed not solved by the existing algorithm need to improve the performance of the cloud resource based on different metrics, such as data center processing time, response time, task transfer cost, resource utilization and so on.

In this study, we proposed a hybrid task scheduling algorithm based on amalgamation of particle swarm optimization and ant colony optimization to assign the users' tasks to multiple computing resources and balance the entire load of the system. The hybrid algorithm has been evaluated and compared with other meta-heuristic algorithms using CloudAnalyst simulator.

The algorithm tested based on the resource information in heterogeneous environment. The result of the study shows that the proposed scheduling algorithm has a better response time and processing time than other basic meta-heuristic algorithm.

Keyword: *Cloud computing, Task scheduling, Anti colony optimization, Particle Swarm optimization, CloudSim, CloudAnalyst.*

CHAPTER ONE

1. INTRODUCTION

1.1. Overview of Cloud Computing

Cloud computing refers to a collaborative Information Technology environment, which is deliberate with the aim of measurable and remotely spreading scalable IT resources for effective and green utilization. Cloud Computing is intended to enable the computing across largely and diverse resources and dynamic stability. It is a subsidized form of cluster computing and utility computing. It has the potential of creating the 'computing as a utility' a reality in future. It describes a large number of applications that are accessible through the internet and for this purpose large data servers to host web applications are used [1]. It provides an on-demand computing that highlights subscription based services where one can obtain huge number of computer resources and the storage spaces. The clouds depict virtualized data centers. Cloud Computing provides scalable, pay-as-you-go, just in time allocation of resources. Cloud Computing provides a new deployment model of resources as accessible using public or private networks. National Institute of Standards and Technology states the cloud computing definition as, "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". Five critical characteristics of cloud computing listed with the aid of NIST are, "on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service".

In cloud computing, there are three service models [2] [3]: Infrastructure as a Service, Platform as a Service, and Software as a Service. IaaS provides access to essential resources such as physical machines, virtual machines, virtual storage. PaaS offers the runtime environment for applications. It also offers development and deployment tools, required to develop applications. SaaS model allows providing software application as a service to the end users. It refers to software that is deployed on a hosted service and is accessible via Internet.

There are four types of deployment models which are [4]: public cloud, private cloud, community cloud and hybrid cloud. **Public cloud:** It is a deployment model which provides system and

services which are easily accessible by public. It may be less secure because of its openness, for example email. **Private cloud**: Private cloud provides systems and services which are manageable within an organization. This type of cloud provides more security because of its privacy. **Community cloud**: It provides system and services which are accessible by group of organizations. In this type of cloud information is confined to the owner of organization. **Hybrid cloud**: Hybrid cloud includes both public and private cloud. Hybrid cloud is capable of providing flexibility in business and some data deployment options.

1.2. Issues in Cloud Computing

There are different issues in cloud computing such as: **Resource Management:** Users are utilizing less of their own existing resources, while increasing usage of cloud resources. With the evolution of new technologies such as mobile devices, these devices are commonly under-utilized, and can provide same functionality to a cloud provided they are properly configured and managed. **Task Scheduling:** is the main problem area in both Grids as well as in cloud computing. Cloud computing is an evolving concept in IT domain. The scheduling of the cloud services to the consumers by service providers reflects the cost benefit of these computing paradigms. **Security:** issues faced by cloud providers and their consumers. **Fault Tolerance:** The increasing demand of cloud computing as an attractive alternative to information processing systems has increased the value of its correct and continuous operation even in the presence of faulty components. **Load Balancing**: Cloud Computing is an emerging computing paradigm. The main aim is to share data, calculations, and service done with a scalable network of nodes. As Cloud computing stores the data and resources in the open environment. So, the huge amount of data storage increases rapidly [11].

Job scheduling in cloud environment is responsible to assign user tasks to appropriate resources for execution. Since the cloud resources are abstracted, virtualized and dynamically accessible. Task scheduling problem has always been a critical issue in cloud environment. In order to improve the task scheduling issue in cloud environment the researcher community proposed a scheduling technique is called task scheduling algorithm. There are two types of task scheduling algorithms: Static task scheduling algorithm and Dynamic task scheduling algorithm. Static task scheduling algorithm is schedule tasks in known environment i.e. it already has the information about complete structure of tasks and mapping of resources before execution, estimates of task

execution/running time. Dynamic task scheduling is should rely upon not only the submitted tasks to cloud environment but also the current states of system and computer machines to make scheduling decision.

Most of the existing algorithms are trying to solve task scheduling problem based on different performance metrics but there is some problems are existed not solved by the existing algorithm such as find overloaded virtual machines and transfers virtual machines task to another low loaded virtual machine as well as need to improve the performance of the cloud resource based on different metrics, such as data center processing time , response time, task transfer cost, resource utilization and so on. Therefore, need to develop effective task scheduling algorithm to overcome the existing scheduling problem.

In this research, a task scheduling algorithm called particle swarm ant colony optimization task scheduling algorithm has been proposed. The proposed algorithm is considered an incorporation of the Particle Swarm Optimization and Ant Colony Optimization algorithms. According to the proposed PSACO task scheduling algorithm; PSO algorithm is prioritize a user request and calculate the best fittest value to allocate the cloud resource. ACO algorithm is used to determine the best path for resource allocation and shares workload for all VMs. The proposed PSACO algorithm is implement using a cloudsim extension called cloud analyst simulator with a different number of independent tasks and different number of VMs and compare the performance relative to the standard PSO, ACO and ABC in terms of completion time, response time and data center processing time.

1.3. Statement of the Problem

Cloud computing is recently a booming area and has been emerging as a commercial reality in the information technology domain. However, the technology is still not fully developed. There are still some areas that are needed to be focused on Resource management and Task scheduling. Task scheduling at the server end is very difficult because the number task requesting is very large in number, which required some resources to execute. The schedule thus build by the server must be optimal and good enough so that each request by the user gets response in time and every task gets proper resources for its execution. Task scheduling is a major problem in cloud environment and it leads to degrade the performance of the resource allocation in dynamic nature. This also makes more difficult to dispatch the task to resource from the queue and one more difficult is in

provisioning or allocating the task in cloud data center. Some of the existing system distributes load nearly uniform among VM's but it can be further optimized by using paradigms of parallel computing. There are several dynamic task scheduling approaches which are available in cloud environment to solve some part of task scheduling and load balancing problem. PSO (Particle Swarm Optimization) algorithm task will assign to the virtual machine randomly by calculating fitness value. I.e. PSO will check all the virtual machine and assigns the task to proper virtual machine which will have least memory wastage. User sends their task request to the cloud server and this cloud server will decide virtual machine to store that task. Cloud server will select the virtual machine based on the particle swarm optimization, but it takes large amount of time with a huge number of tasks. The convergence speed of PSO algorithm is fast in the initial stage, but the local searching ability will decrease in the later due to the lack of population diversity. ACO (Ant Colony Optimization) algorithm task will assign to the virtual machine in FCFS (First Come First Served) manner. User sends request to the cloud server and this cloud server will select the shortest path to assign the task to a proper virtual machine. But this algorithm does not check the current status of VMs to balance the load when there is an overloaded in virtual machine and there is a waiting time due to FCFS manner i.e. doesn't consider any other criteria for scheduling. ACO algorithm has good optimization ability and the early convergence speed is slow due to the lack of pheromone in initial stage.

1.4. Research Question

- How to improve performance of particle swarm optimization, ant colony optimization and artificial bee colony task scheduling algorithm based on different metrics by developing a new task scheduling algorithm?

1.5. Purpose of the Study

The purpose of the study was to establish an efficient task scheduling mechanism which utilizes particle swarm ant colony optimization (PSACO) task scheduling algorithms to balance the entire load of the system and schedule the tasks among the cloud resource in cloud computing and also to improve the performance of the existing task scheduling algorithm by implementing PSACO task scheduling algorithms based on different metrics such as, response time, data center processing time and transfer cost.

1.6. Objectives

1.6.1. General Objective

The main objective of this research is to implement a particle swarm ant colony optimization task scheduling algorithm to assign the users' tasks to multiple computing resources and to compare the result of proposed algorithm with the basic meta heuristic task scheduling algorithms.

1.6.2. Specific Objective

- Create a complete literature review related to the task scheduling.
- Design a task scheduling algorithm based on the characteristics of particle swarm optimization and ant colony optimization.
- Configure a cloudsim and cloud analyst simulation environment tool to test the implementation of the proposed algorithm
- Implement the proposed algorithm in cloud simulator.
- Evaluate the performance of the proposed algorithm based on response time, transfer cost and data center processing time metrics.
- Compare the proposed algorithm with the existing task scheduling algorithm.

1.7. Scope and Limitations

The scope of this research is to investigate the basic meta heuristic task scheduling algorithms i.e., PSO, ACO, ABC and GA and find their gap to propose new task scheduling algorithm which can solve the problem of current scheduling algorithm. Three basic Meta heuristic task scheduling algorithms (i.e. particle swarm optimization, ant colony optimization and artificial bee colony) were selected for benchmarking and evaluation purpose.

The task scheduling algorithms were developing under data center level. It only manages the user request arrived on the data center and assign the requests based on the current performance of VMs. This research only concerns to achieve optimal scheduling of the user tasks and balancing the entire load of the virtual machine.it doesn’t include other cloud computing issue. The propose algorithm implemented and test in cloud simulation tool, it doesn’t consider real deployment environment due to the requirement of more time and resource.

1.8. Significance of the Study

This research study focus towards to improve the performance of PSO, ACO, ABC and GA task scheduling algorithm in cloud computing. Cloud users can access a cloud resource with in a minimum time and pay as per use. The cloud providers can easily manage a cloud resource and they become more cost effective due to optimal scheduling of a task and balancing the entire load of the system. The proposed algorithm can be used to achieve quality of services and minimize energy consumption due to task migration from overloaded virtual machines to underloaded virtual machines.

Furthermore, the research could be used as a reference point for further research and studies for new task scheduling and load balancing algorithm strategies.

1.9. Organization of the Thesis

The research consists of six main chapters; the rest of the research is organized as follows: Chapter Two is about the detailed literature review about task scheduling algorithms in cloud computing environment. This section includes, the architecture of cloud computing and task scheduling algorithms, different types of task scheduling algorithms for improved quality of service in the cloud and some related and recent research works.

Chapter Three present about the methodology of the research and strategy to identify task scheduling algorithms solutions. In this section describe about research design, data collection technique, sampling techniques, data sources, cloud computing tools and issues of reliability and validity.

Chapter Four presents about the proposed solution which is called particle swarm ant colony task scheduling algorithm in cloud computing environment.

Chapter Five present about the experimental design and implementation of the proposed algorithm, the simulation configuration and the experimental results of the proposed algorithm and existing algorithm.

Chapter six is the last chapter which is for conclusions and future works.

CHAPTER TWO

2. LITERATURE REVIEW

2.1. Cloud Computing

Cloud computing is getting progress constantly as a new type of computing. Cloud computing provides a way to use and access multiple server based computational resources via a digital network internet connection using www. Cloud users can access the server resources using a computer, net book, pad computers, smart phone or other devices. Cloud computing applications are managed and provided by the cloud server. In cloud computing data is also stored remotely in the cloud configuration. Cloud computing provides a way to delivery of computing resources over the internet. When we store pictures and video online use webmail or a social networking site we use cloud computing services. Cloud computing makes IT management easier and more responsive and it is cost effective to the changing needs of the business [4].

According to [12] explanation, Cloud computing is an evolving concept which makes better use of multiple distributed resources that can be allocated to users as per requirements. This helps in cheaper and efficient utilization of available resources and easier handling of larger computational problems. Its advantages include transparency of resources, flexibility, location independence, reliability, affordability, and greater availability of services, etc. To provide these facilities, the tasks need to be scheduled properly on the resources so as to provide maximum performance in minimum time.

This resource allocation process is performed in two stages. In the first stage, the load balancer allocates resources to systems as requested by an application. The second stage takes place when incoming requests are assigned to an application to balance loads within the application as per Quality of Services (QoS) and minimum cost [36]. To minimize the total time taken, the scheduling principle should aim to reduce the amount of data transfer with minimum cost and ensure balanced distribution of tasks as per processing capability [37].

As [13] explanation, Cloud computing is a rising technology in distributed computing which facilitate pay per model as per user command and requirement. Cloud consists of a set of virtual machines which includes both computational and storage facility. Now a day, cloud computing

technology is used all over. Cloud provides many facilities due to its vast area such that sharing of resources for different purposes. The primary aim of cloud computing is to give most suitable access to remote scattered resources.

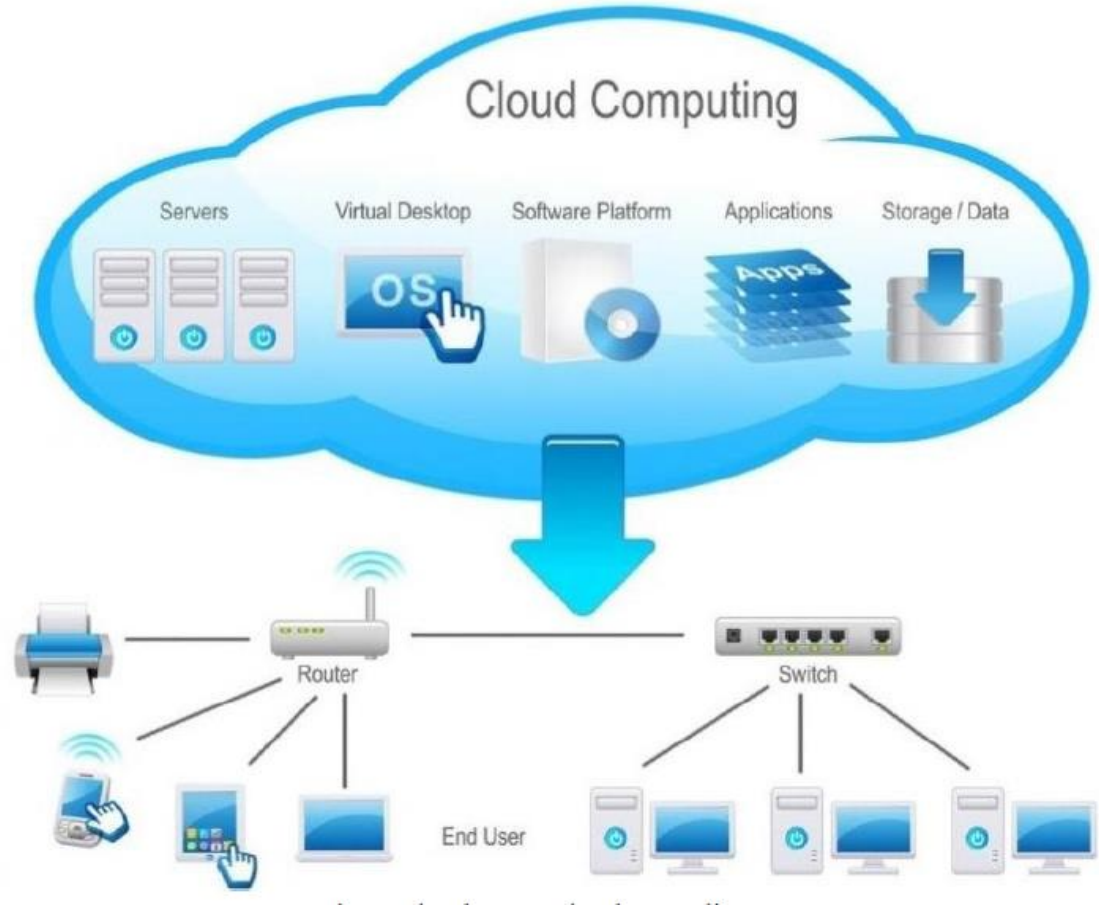


Figure 2.1: Cloud computing layout diagram [13]

According to [5] explanation, Cloud computing is considered the latest network infrastructure that supports the decentralization of computing. The main features of the Cloud are the possibilities for building applications and providing various services to the end user by virtualization on the internet. One of the foremost challenges in the field of the cloud computing is the task scheduling problem. Task scheduling problem concerns about the dynamic distribution of the tasks over the Cloud resources to achieve the best results.

2.2. Cloud Computing Architecture

Cloud computing architecture typically consists of a front end and a back end connected by Internet or Intranet networks. The front end comprises of client devices which can be thin client, fat client or mobile devices. The clients need some interface and applications for accessing the cloud computing system. The back end consists of the various servers and data storage systems. A central server is typically used for administering the cloud system which includes monitoring the overall traffic and fulfilling the client demands in an efficient manner. Cloud computing is a new form of distributed computing that delivers infrastructure, platform, software and applications as services. Cloud generally provides three levels of services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [14].

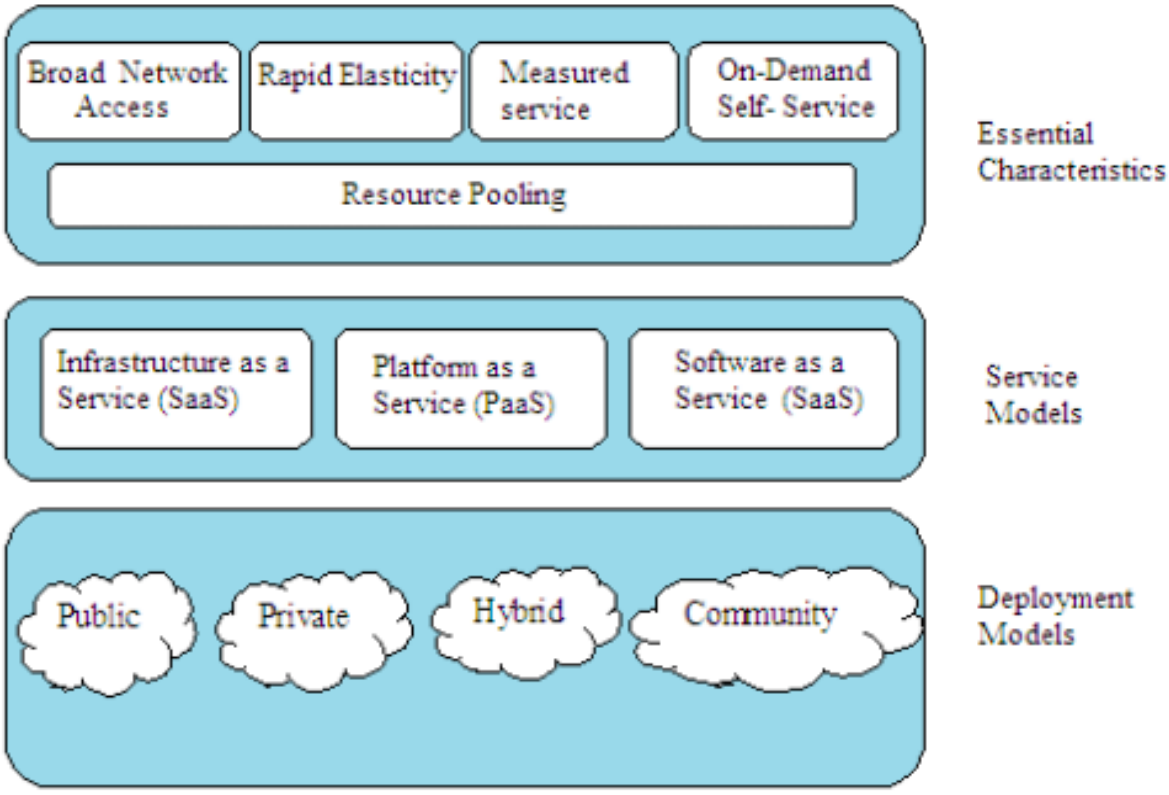


Figure 2.2: Architecture of Cloud Computing [4]

Deployment Models are classified just as Public Clouds, Private Clouds, Community Clouds and Hybrid Clouds. This huge combination of services and sources, shared through clients on subscription basis needs a serious attention in terms of tasks scheduling, resource allocation and resource sharing.

2.3. Characteristics of Cloud Computing

National Institute of Standards and Technology states the cloud computing definition as: - "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". Five critical characteristics of cloud computing listed with the aid of NIST are, "on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service".

On –demand

Computer values like email, applications, network or server service can be presented without requiring human cooperation with each service provider. Cloud service providers providing on demand self-services comprises Amazon Web Servicing (AWS), MS, Google and International Business Machines (IBM and Salesforce.com. New York Times and NASDAQ are examples of companies using AWS (National Institutes of Standards and Technology). Gartner defines this characteristic just as service based.

Broad network connection

Cloud capabilities are accessible over the network and accessed through normal mechanisms that further use by heterogeneous thin or thick client platforms such as mobile phones, laptops and Personal Data Assistant.

Resource Pooling

The provider's computing resources are pooled together to serve many consumers using multiple-tenant model, with distinct physical and virtual resources dynamically assigned and reassigned according to customer requirement. The resources consist of amongst others storage, processing, memory, network radio bandwidth, virtual machines and email services. The pooling together of the resource builds economies of scale.

Rapid Elasticity

Cloud services could be quickly and elastically provisioned, in some cases automatically, to quickly scale out and quickly released to instantly scale in. To the consumer, the capabilities available for provisioning often appear to be huge and can be purchased in any quantity at any time.

Measured service

Their source usage can be measured, composed, and noted given that transparency for both the provider and consumer of the utilized service. Cloud computing services apply a metering ability which enables to control and optimize resource use. It is just related to air time; electricity or municipality water IT services are owed per usage metrics – pay per use.

2.4. Virtualization in cloud computing

Cloud has an extra layer called virtualization layer. This layer acts as a creation, execution, management and hosting environment for application services [39]. Virtualization is not exist in real, but it provides everything like real. Virtualization is the software implementation of a machine which will execute different programs like a real machine. Through the virtualization consumer can use the exclusive programs or services of the cloud, so this is the main a part of the cloud environment.

Two types of virtualization are [15]:

- **Full Virtualization**: Full virtualization means a complete machine is installed on another machine. That virtual machine provides all the function which exists on the original machine. It facilities when actual machine not free then user use the virtual machine.
- **Para virtualization:** Para virtualization means the hardware allows multiple operating systems to run on single machine. It also allows efficient use of system resources such as memory and processor.

As [19] explanation, Cloud computing uses virtualization technique for mapping the resources of cloud to the virtual machine layer, implement the user's task, so the task scheduling of cloud computing environment achieves at the applications layer and the virtual layer of resources. Task scheduling in cloud computing

Based on [16] research paper, the task scheduling goals of cloud computing is to provide optimal tasks scheduling for users and provide system throughput and QoS at the same time. Specific goals are load balance, QoS, minimum cost, and the optimal operation time and system throughput. In the Cloud Computing Systems, the number of users increased the tasks to be scheduled in Cloud increased proportionally. Therefore, there is a need for better algorithms to schedule tasks on these systems [17].

According to [38] explanation, the main target of scheduling is to maximize the resource utilization and minimize processing time of the tasks. The scheduler should order the tasks so that balance between improving the quality of services and at the same time maintaining the efficiency and fairness among the tasks. An effective task scheduling strategy must goal to yield less response time so that, the execution of submitted tasks takes place within a deadline as a consequence of this, tasks takes place and more number of tasks can be submitted to the cloud by the users which results in enhancement of the performance of the cloud system [18].

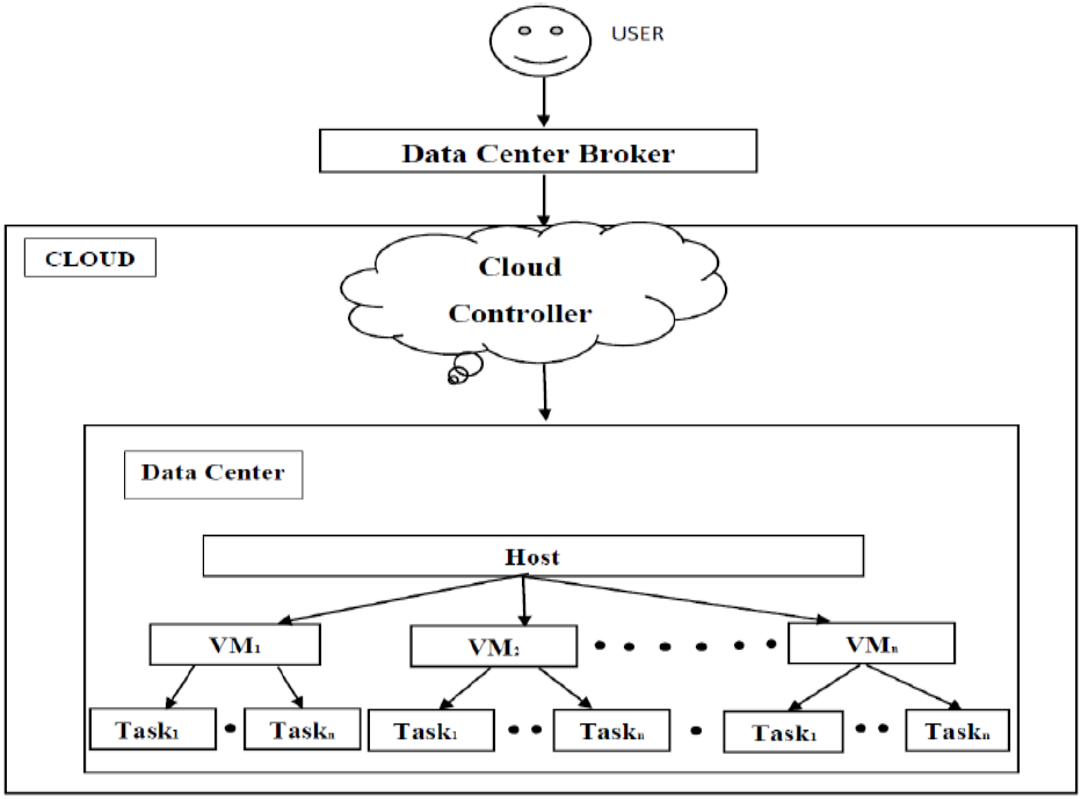


Figure 2.3: Scheduling Architecture [18]

In Scheduling Architecture, Users submit their tasks to Data Center Broker, this broker behaves like a dispatcher between user and Data center and helps to schedule task on virtual machines. In Data Centre, there is number of host on which number of virtual machines are scheduled and on those VM task are scheduled according to the scheduling polices taken by Data Centre Broker. The number Data Centre Broker is similar to the number of users of cloud. The Data Centre Broker communicates with cloud controller and schedules the submitted tasks.

Task scheduling is a significant research topic whose objective is to ensure that every computing resource is distributed efficiently and fairly and in the end, improves resource utility. In traditional computing environments of distributed computing, parallel computing and grid computing, a set of static and dynamic and mixed scheduling strategies have proposed.

There are two types of **task scheduling algorithms** [19]: -

1. **Static task scheduling** schedule tasks in known environment i.e. it already has the information about complete structure of tasks and mapping of resources before execution, estimates of task execution/running time.
2. **Dynamic task scheduling** must depend on not only the submitted tasks to cloud environment but also the current states of system and computer machines to make scheduling decision

2.5. Task scheduling algorithms

There are different task scheduling algorithms are existing for solving complex problems in cloud computing environment. Such as: Genetic algorithm, Particle swarm optimization, Ant colony optimization and artificial bee colony are the most well-known heuristics algorithms used for efficient scheduling of tasks to cloud resources.

2.5.1. Genetic Algorithm

Genetic Algorithm was the first evolutionary technique that is based on the principle of natural selection and Mendel's laws of inheritance. GA has various advantages over other techniques for computationally intensive problems, if provided with properly set operators and fitness functions. GA defines a set of solutions (chromosomes) that are collectively called a population. The method then performs crossover, mutation and selection operations iteratively till the stopping criteria is satisfied [12].

Genetic algorithm is useful in solving variety of optimization problems. GA is basically applied to dynamic scheduling problems with multiple tasks. These task must be non-identical, autonomous numerous task. They must be distributed in shared multiprocessor memory system. Structural Genetic Algorithm provides the option of solving the solution structure solution parameter problems with accuracy [20].

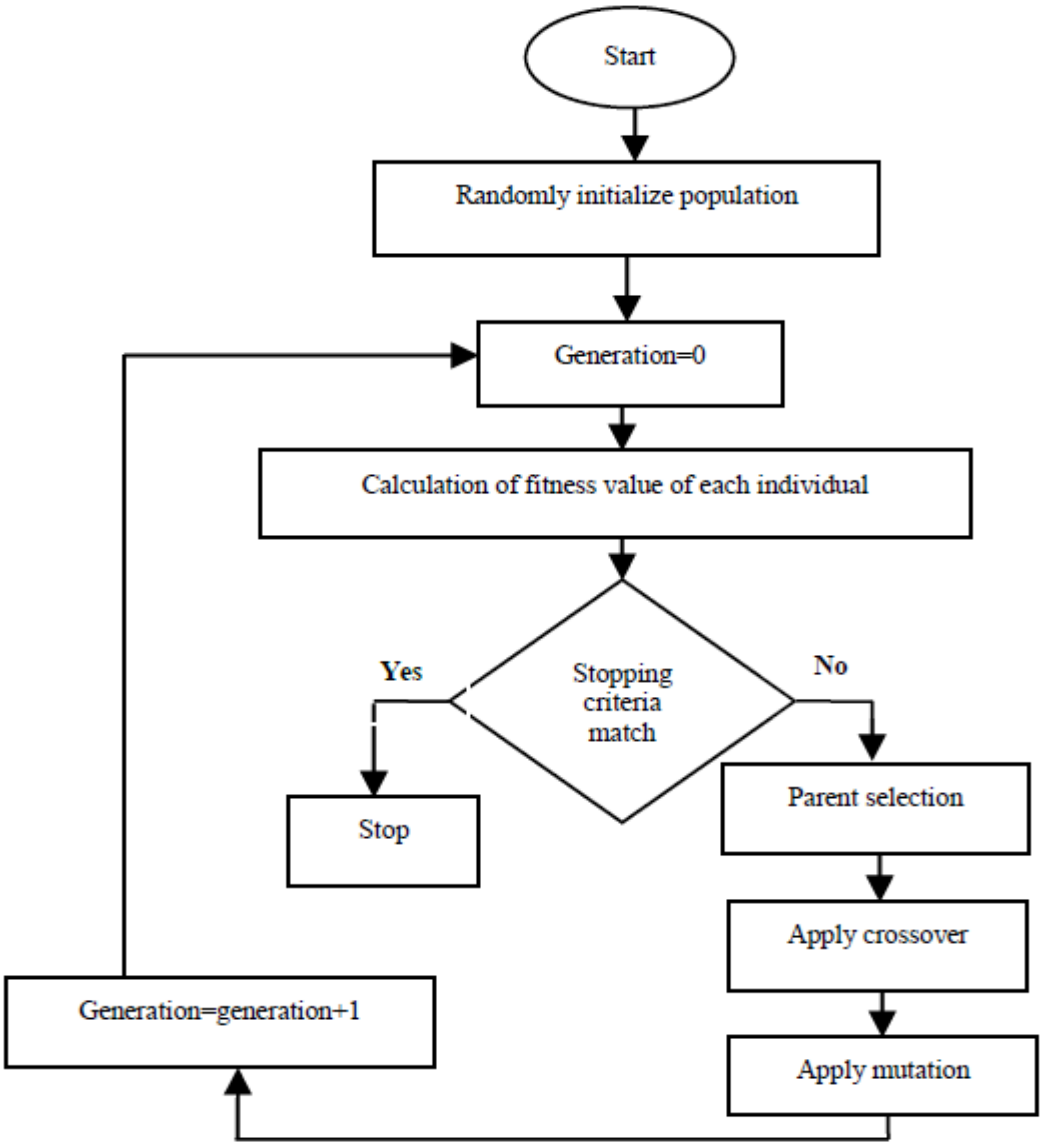


Figure 2.4: Flow chart for genetic algorithm [20]

GA is implemented by the following steps [20]:

1. A population with randomly generated individuals (chromosome) is taken.
2. The fitness function for each and every individual is calculated.
3. Two chromosomes selected, as parents which has best fitness value.
4. Crossover between the parents is applied with probability and crossover rate.
5. Mutation is applied with probability and mutation rate.
6. Repeat Step 3- Step 6, until enough members are generated.

GA has complexity and longtime consumption for huge number of tasks. It takes time for convergence.

2.5.2. Particle Swarm Optimization

PSO has recently emerged as a prominent heuristic approach, applicable to various large and complex problems, like task scheduling problem, knowledge extraction in data mining, electric power systems, etc. PSO follows the principle of random searching in entire solution space using a large population, depending upon the problem domain [12].

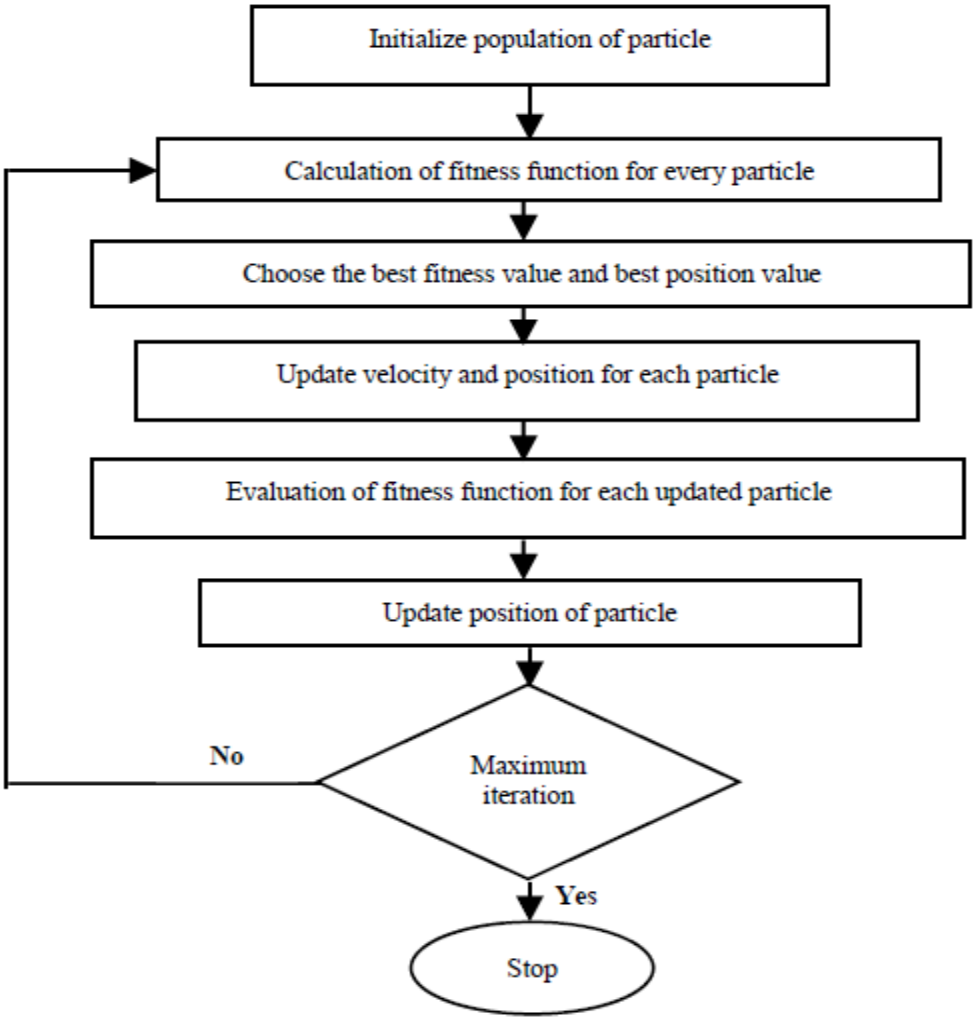


Figure 2.5: Flow chart for particle swarm optimization [20]

PSO has been widely accepted as a global optimization algorithm of current interest for distributed optimization and control [35]. PSO algorithm assign task to the virtual machine in the manner of best fit. Here task check the virtual machine and task is assigned to the proper virtual machine that has minimum wastage of memory. PSO is used to scheduling applications in cloud resources that take transmission cost and computation cost into account [20].

PSO is implemented by the following steps [20]:

1. Initialize population of particle.
2. The fitness function is calculated for every particle.
3. The current fitness value of the particle's is compared with other particle's fitness value and finds P_b (local best) value.
4. Compare the overall previous best value of population and fitness value to obtain G_b (global best).
5. By using the following (1) and (2) equations, particle's position and velocity is update.
6. If the number of equation is reached maximum, then stop otherwise repeat from step 2.

$$V_t^{id} = V_t^{id} + c_1 r_1 (P_t^{id} - S_t^{id}) + c_2 r_2 (P_{gd}^{id} - S_t^{id}) \quad (1)$$

$$S_t^{id+1} = S_t^{id} + V_t^{id+1} \quad (2)$$

Where, the velocity for particle t is V_t^{id} and position of the particle t is S_t^{id}. Index of the particle is *t*. *r1*, *r2* are arbitrary numbers, belongs to (0, 1) and *c1*, *c2* are random variable. P_t^{id} is the local best position of t^{th} particle and P_{gd}^{id} is the global best position of t^{th} particle.

2.5.3. Ant Colony Optimization

ACO is a meta-heuristic method inspired from models of cooperative food search in ants. A set of agents is used to implement the behavior of real ant colonies where ants cooperate and communicate through pheromone trails. An ant solves a problem iteratively by using a construction graph where edges represent the possible partial solution that the ant can take according to a probabilistic state transition rule. After selecting a partial or a complete solution, a rule of pheromone updating starts. This rule gives a feedback mechanism to speed up convergence, and prevents premature solution stagnation. Due to the elaborate characteristics of ACO, various algorithms based on the ACO meta-heuristic have been applied to many difficult optimization problems [12].

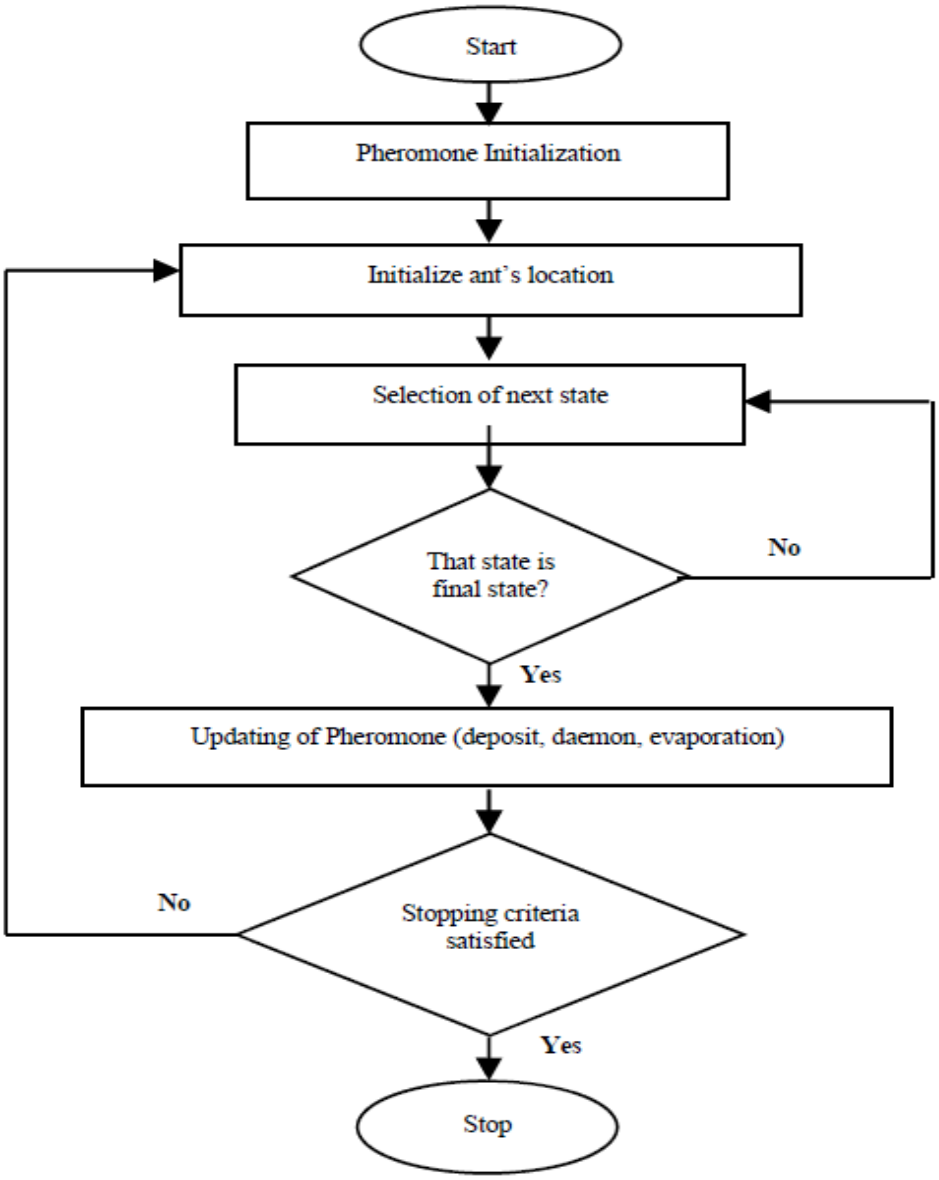


Figure 2.6: Flow chart for ant colony optimization [20]

ACO is implemented by the following steps [20]:

1. First Pheromone Initialization.
2. Location of the ant is initialized as an entry state.
3. Next state will be selected
4. Check if the next state is final state or not if it not then repeats from state 3, if yes the do state 5.
5. Pheromone updating step (deposit, daemon and evaporate of pheromone).
6. If stopping criteria is satisfied than stop the execution, else repeat from step 2.

2.5.4. Artificial Bee Colony

The ABC algorithm is based on the activities of bees while searching for nectar, and sharing the information with other bees. There are three types of agents: - the employed bee, the onlooker bee, and the scout. The employed bee stays on a food source and provides its surroundings in memory; the onlooker acquires this data from the employed bees and selects one of the food sources to forage; and the scout performs the task of finding new nectar sources [12].

Artificial bee colony is used for numerical optimization, combinatorial optimization, and unconstrained and constrained optimization problems. It also provides a good explanation to MR (Magnetic Resonance) brain image classification and inference of face pose [20]

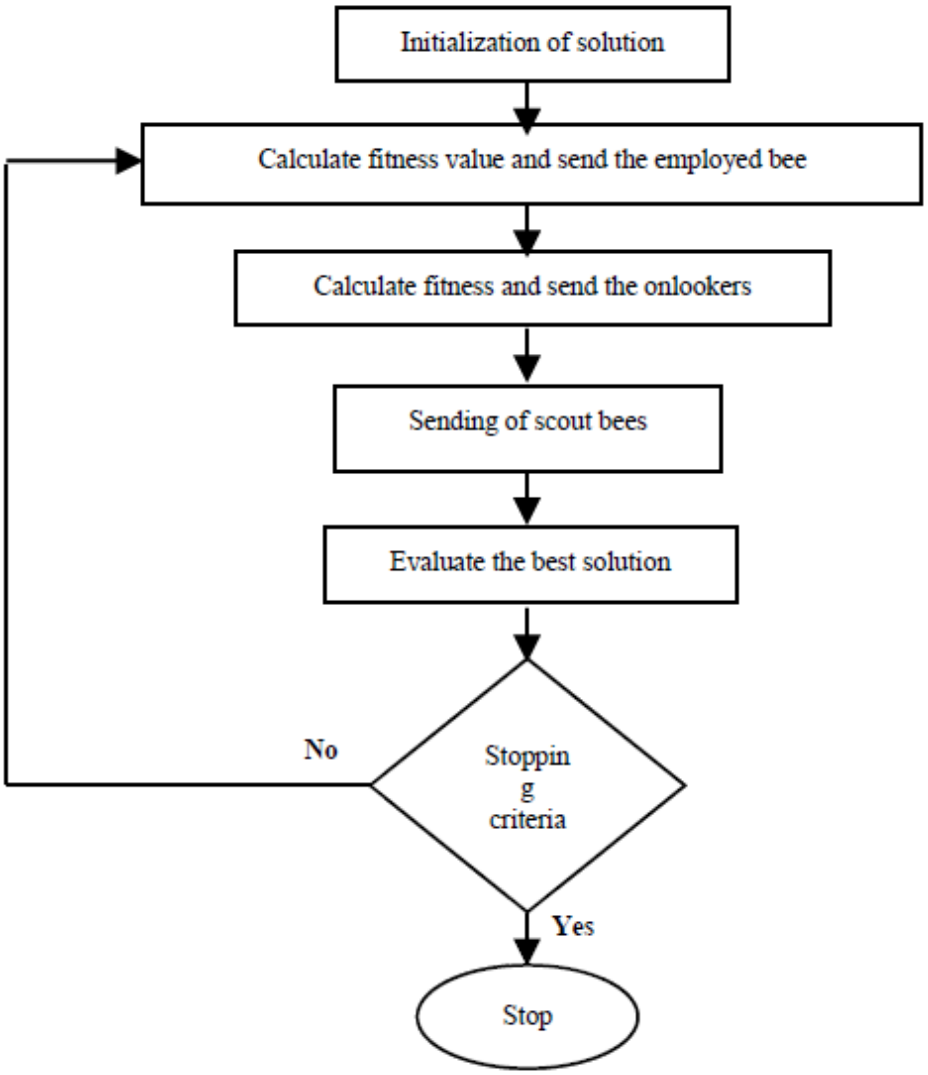


Figure 2.7: Flow chart for artificial bee colony [20]

ABC is implemented by the following steps [20]:

1. Control parameter initialization.
2. Solution initialization.
3. Calculate the fitness value and employed bee sending.
4. Calculate the fitness value and the onlooker's bee sending.
5. Send the Scout bees.
6. Evaluate the best solution.
7. Stopping criteria checking if match then stop, otherwise repeat from 3

2.6. Related Work

Based on [7] research paper, "task scheduling algorithm based on load balancing using Artificial Bee colony algorithm ABC for cloud computing environments" is proposed to achieve well balanced load across virtual machines and deliver to the minimum makespan. ABC algorithm is one of the most popular bio-inspired techniques. Artificial bee colony algorithm is an optimization metaheuristic algorithm based on an intelligent behavior of honey bee swarms. Evaluate the proposed algorithm in different condition and dataset. The proposed approach use cloudsim toolkit for simulating. The experimental result shows the performance of proposed algorithm is better than the ACO algorithm and FCFS algorithm. The ABC algorithm has slow convergence rate in later stage and don't consider QOS.

In the research paper [10], proposed a "Cloud Task Scheduling Based on Ant Colony Optimization" to find the optimal resource allocation for tasks in the dynamic cloud system. The aim of the proposed algorithm is minimizing the makespan of a given tasks set. ACO algorithm is random optimization search approach that will be used for allocating the incoming jobs to the virtual machines. A cloud task scheduling policy based on ACO algorithm compared with different scheduling algorithms First Come First Served and Round-Robin. Algorithms have been simulated using cloudsim toolkit package. The experimental results showed that cloud task scheduling based on ACO outperformed FCFS and RR algorithms

According to [5], proposed a task scheduling algorithm for cloud computing based on a merge PSO algorithm with the Cuckoo Search algorithm, called (PSOCS). To evaluate the proposed algorithm, the cloudsim simulator tool has been used. The experimental results show the reduction of the makespan and increase the utilization ratio of the proposed PSOCS algorithm compared

with PSO algorithms and Random Allocation. The propose PSOCS algorithm need an improvement based on other different metrics in order minimum execute time, and maximum of utilization resource.

Based on [6] paper, “Artificial Bee Colony with longest job first (LJF)” is proposed to optimize the scheduling of Virtual Machine (VM) on Cloud computing. The proposed algorithms analyze the difference of VM load balancing algorithm and reduce the makespan of data processing time based on different parameter setting. In the experiment compare ABC_FCFS, ABC_SJF and ABC_LJF with FCFS, SJF and LJF. The experimental result showed that ABC_LJF performed a good performance scheduling strategy in cloud environment and reduces the makespan of data processing time than other algorithms. The proposed algorithm is not considering if the VMs are overload and only schedule one task at a time. Therefore, high consumption of power due virtual machine overloaded and maximum waiting time

In research paper [8], proposed a “Round Robin VM Load balancing algorithms”. In available Virtual Machine Load balancing policies problem of cloud is that they don’t save the state of the previous allocation of virtual machine to a request from the user and the VM Load balancing algorithm requires execution each time a new request for VM allocation received from user. The proposed Round Robin VM Load balancing algorithms is save the state of the previous allocation of virtual machine to a request from the user. The Round Robin VM Load balancing algorithm implemented using java language in Cloudsim toolkit and compare with the existing round robin method. Experimental result shows that the proposed algorithm improves the performance by consuming less time for scheduling virtual machine. The performance of proposed algorithm was evaluated with only Round-robin algorithm and is not consider overloaded and under loaded virtual machines to improve power consumption

In [9], proposed “Bi-Criteria Priority based Particle Swarm Optimization Workflow Scheduling Algorithm for Cloud (BPSO)”. This proposed algorithm is an approach which is develops for scheduling workflow tasks over the available resources of cloud that minimized the execution time and execution cost under the given deadline and budget constraint. In, Bi-Criteria Priority based Particle Swarm Optimization the workflow tasks are executed in order of their priority which is basically computed using bottom level. The assigned priority is then used to initialize the PSO.

After assigning the priority the tasks are sorted according to the descending order of bottom level. The tasks are then sending to different processors according to their order of execution for completing the workflow application. For performing the experiment author developed a simulation program in java which consists of a data center which includes six resources with different processing speed. Experiment results shows that this algorithm has a promising performance when compared with PSO.

According to [30] "Efficient Optimal Algorithm of Task Scheduling in Cloud Computing environment" is proposed**.** The objective of this paper is to develop a Generalized Priority algorithm for efficient execution of task. Priority is an important issue of job scheduling in cloud environments. The propose algorithm is implemented and tested using Cloudsim toolkit and compare the experimental result with RR and FCFS algorithms. Result shows that the proposed algorithm is more efficient than FCFS and Round Robin algorithm. The proposed algorithm developed with limited tasks; need an improvement for large number of task. In this algorithm is not consider overloaded and under loaded virtual machines to improve power consumption. The algorithm need to improve based on other parameters such as response time, datacenter processing time and so on.

As per [31] research paper, "Genetic-Based Task Scheduling Algorithm in Cloud Computing Environment" is proposed. The aim of this paper is to develop a task scheduling algorithm in the Cloud computing environment based on GA for allocating and executing independent tasks to improve task completion time, minimize the execution cost, as well as, maximize resource utilization. The performance of algorithms evaluated using the Cloudsim- toolkit and compare with RR and default GA based on completion time, cost, resource utilization, speedup, and efficiency parameters. The result shows that the proposed TS-GA algorithm has a better performance than RR and GA. The TSGA algorithm has complexity and longtime consumption for huge number of tasks.

CHAPTER THREE

3. METHODOLOGY OF THE RESEARCH

The aim of this chapter is to design the methodology used to accomplish the objectives of this study which introduced in the previous chapter. The research methodology used to explain how the process of this study is done and systematically try to find the resolution of a problem or a great understanding of a phenomenon.

3.1. Research Design

Research is described as an approach to promote knowledge enhancement. Research design is based on the concept of a designer try to find a solution related to human problem. The creation of innovative artifacts is an important part of the process, whereas the designed artifacts are useful and fundamental in understanding the problem. Artifacts are categorized as several components, including constructs, models (abstractions), methods (algorithms and practices), and instantiations (implemented and prototype systems).

In our research study focus on descriptive research type which describe systematically problem, program and characteristics of an individual situation. Two type of research methods: Qualitative research method and Quantitative research method. In our study, we used both quantitative and qualitative research methods which describe how a new algorithm is expected to support solutions to problems.

Quantitative research utilizes numerical analysis and tries to illustrate the relationship among issues in this study. Qualitative research deals with the description and understanding of the situation behind the issues. Whereas quantitative research often deals with surveys or experiments, qualitative research can use procedures such as field studies.

Most of the time the original reports found in academic journals. And also, the detailing of the research methodology used in-depth descriptions of the research.

The activities of research design begin with problem formulation and move on with suggestions how to solve it. Development and Evaluation is proceeding, with the possibility of moving back

to the problem formulation stage. Finally, a conclusion is drawn where the results are the final output [42].

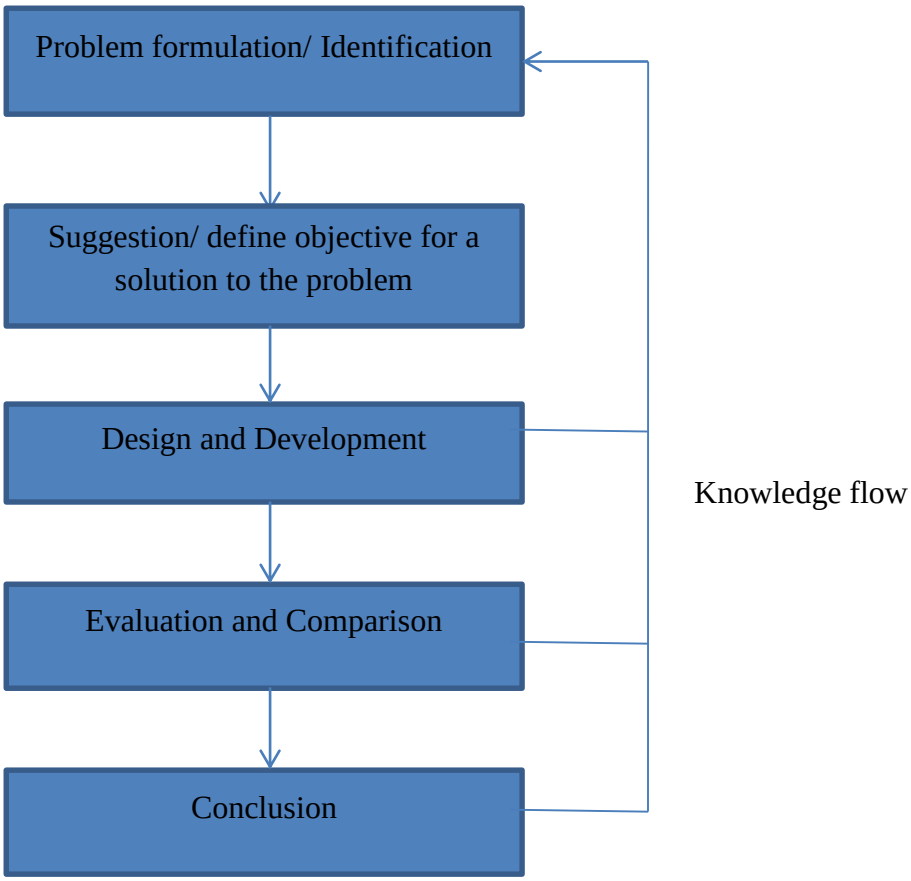


Figure 3.1: Research Design Cycle

Problem formulation / Identification

The problem identification is used as requirements for the development of an algorithm. Reading similar and relevant research published in conference proceedings and journals are essential, which presented in the Chapter 2 (Literature Review).

Review of several algorithms which focus on task scheduling in cloud computing. These reviewed algorithms have also been used in other research areas such as load balancing, grid Computing, distributed computing etc. It is already known that task assignment, load balancing has been found to be NP-complete problems. Clearly, identify the gap in the existing task scheduling algorithms and the parameter used for evaluation techniques.

Objective for solution

The objective of our study address the tasks are needed to solve the problem of the research. Objective can be quantitative how the current one better than the existing one or qualitative how the new algorithm is expected to support solution to the problem not yet addressed.

Design and Development

In this activity develop a unique architecture design, for extensibility, modularity and define functionalities of system components and interrelationships among them. After determining architecture of the new algorithm with its functionality, implement the proposed algorithm in cloud analyst toolkit which runs Eclipse IDE with a different number of independent tasks and different number of VMs

Evaluation and Comparison

After understanding the simulation tool then implement the proposed scheduling algorithm and evaluate the performance based on different metrics such as response time, cost and processing time. Make a comparison of the proposed algorithm performance with the basic Meta heuristics task scheduling algorithm. Analyze and discuss the experimental result of the proposed task scheduling algorithm and the existing scheduling algorithm.

3.2. Data Source

For the study purpose both primary and secondary data are used. In this research, mostly we used secondary data as data source for the study such as, articles, journals, research and other topics related to task scheduling.

- Articles\ Full text research's from early 20's to current year 2016/17 were randomly selected based on relevancy to the research
- The sources were from Google Scholar search engine and various research online databases.

In this study, the data source is the international network. That means the data collected from Google scholar search engine which is the internet user's dataset in 6 geographical locations in the real world. The internet users of the sample are shown for the following table for the study purpose.

Table 3.1: Internet users, Source: (Nielsen Online, www.internetworldstats.com, June 30, 2017)

World Regions	Population (2017 Est.)	Population % of World	Internet Users 30 June 2017	Internet Users %
Africa	1,246,504,865	16.6 %	388,376,491	10.0 %
Asia	4,148,177,672	55.2 %	1,938,075,631	49.7 %
Europe	822,710,362	10.9 %	659,634,487	17.0 %
America	1,010,828,651	13.4%	724,328,531	18.6%
Middle East	250,327,574	3.3 %	146,972,123	3.8 %
Oceania / Australia	40,479,846	0.5 %	28,180,356	0.7 %
World Total	7,519,028,970	100.0 %	3,885,567,619	100.0 %

Within the internet user the data set organized in the form of structured and unstructured data. In structured data, the dataset will contain the same structure and searchable by a data type. Whereas, in unstructured data the dataset will not contain the same structure and depending on software program each individual data may contain its own specific structure or format. Images, videos, email, documents and text are all considered to be that of unstructured data.

I. Primary Data

Primary data are the data collected by a researcher specifically for a research assignment. In other word, information that an organization must gather because no one has compiled and published the information in a forum accessible to the public. Organizations generally take the time and allocate the resources required to gather. In this type of data gathering method, the data which the researcher collects through various methods such as interviews, surveys, questionnaires, focus groups etc. It can be also collected across the national borders through emails and posts. It can include a large population and wide geographical coverage.

The primary data are unique and relevant to the topic of the research study so the degree of accuracy is very high. Moreover, primary data is current and it can better give a realistic view to the researcher about the topic under consideration. Reliability of primary data is very high because

these are collected by the concerned and reliable party. In this study, we collect relevant data through emails from other researchers who have a skill related with this research idea.

II. Secondary Data

Secondary data are the data collected by a party not related to the research study but collected these data for some other purpose and at different time in the past. In our research study this data become secondary data for the current users. These may be available in written, typed or in electronic forms. A variety of secondary information sources is available to gather data on an industry, potential product applications and the market place. Secondary data is also used to gain initial perception into the research problem and classified in terms of its source either internal or external. Internal, or in-house data, is secondary information acquired within the organization where research is being carried out. External secondary data is obtained from outside sources.

Secondary data is that it is cheaper and faster to access. It provides a way to access the work of the best scholars all over the world. Secondary data save time, efforts and money and add to the value of the research study.

3.3. Data Collection Techniques

The data collection is gathering of data for a particular purpose from various sources and which is used to explain about the process of the collecting data. The information or data that can comes from the range of the sources. To collect the primary data, we use different types of techniques. For instance, in our research we used googling or action search techniques to collect data from the internet. Data collection is the process of gathering and measuring the information or data on objected variables in the established systematic fashion. And also, it enables a person or organization in order to answer relevant question and to evaluate the outcomes and make predictions about the future probabilities and trends. In the research study the data collection is defined as the systematic collection, analysis and interpretation of the necessary data to design implement and evaluate. It is also used to develop effective strategies.

3.4. Issues of Reliability and Validity

Evaluating the quality of research is essential if findings are to be utilized in practice and incorporated into care delivery. Validity is the precision in which the findings accurately reflect the data. Reliability is the consistency of the analytical procedures including accounting for

personal and research method biases that may have influenced the findings. Generalizability is the transferability of the findings to other settings and applicability in other contexts [21].

In our research, we used different validation criteria and compare the result in terms of overall response time, data center processing time and others. The quality of research defined depending on reliability and validity and measures the performance of a proposed scheduling algorithm using the cloudsim tool.

3.5. Sampling Techniques

Sampling is selecting a suitable sample or a representative part of a population for the purpose of determining parameters or characteristics of the whole population [22]. We used a smaller number of items/parts of among the larger population to make conclusion about the entire population. For our simulation let assume 0.0001% of population use internet during peak time from the total number of internet users and off peak time users is $1/10^{th}$ of peak time users. In this study, the sampling population of the research includes any active internet users of the selected sex (6) world regions. Assume here one UB represents as thousands of peoples in the regions.

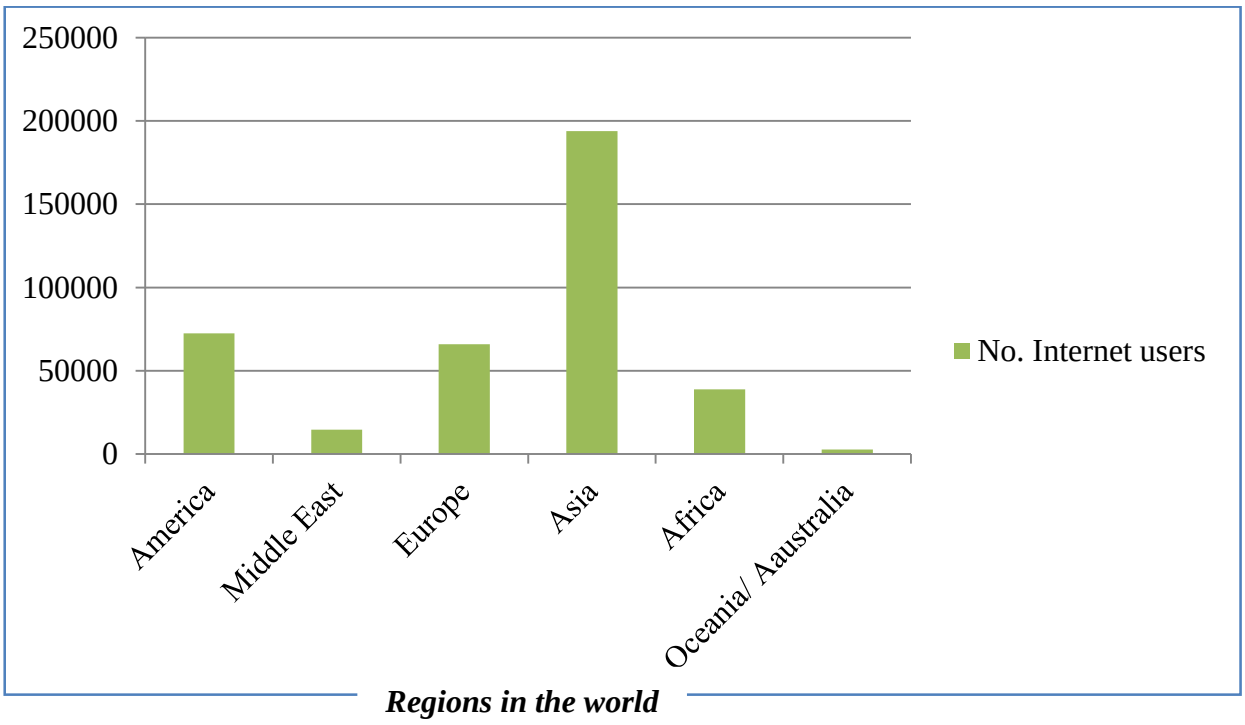


Figure3.2: Internet user in 2017. Source: (Nielsen,www.internetworldstats.com, June 30, 2017)

3.6. Cloud Computing Tools

The proposed approach is tested using ***Cloud Simulator*** tool which runs on eclipse IDE with a different number of independent tasks and different number of VMs.

I. Cloudsim

In [23] explanation, cloudsim is a framework developed by the GRIDS laboratory of University of Melbourne which enables seamless modeling, simulation and experimenting on designing cloud computing infrastructures. Cloudsim is a self-contained platform which can be used to model DCs, service brokers, scheduling and allocation policies of a large scaled cloud platform. It provides a virtualization engine with extensive structures for modeling the creation and life cycle management of virtual engines in a DC. CloudSim framework is constructed on top of gridsim framework also developed by the GRIDS laboratory [32].

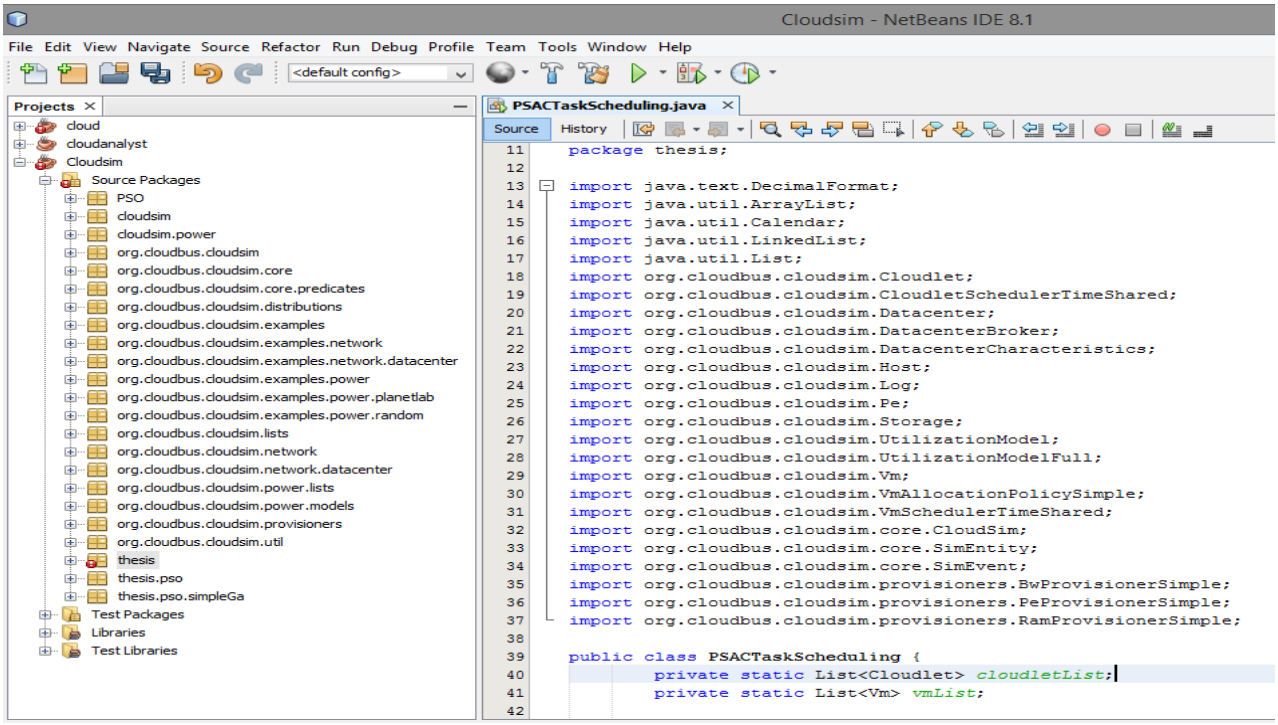


Figure 3.3: Cloudsim sample simulation screen

II. GridSim

GridSim toolkit was developed by Buyya et al to address the problem of near impossibility of performance evaluation of real large scaled distributed environments (typically Grid systems but also P2P networks) in a repeatable and controlled manner. The GridSim toolkit is a java based simulation toolkit that supports modeling and simulation of heterogeneous Grid resources and

users spread across multiple organizations with their own policies. It supports multiple application models and provides primitives for creation of application tasks, mapping of tasks to resources and managing such tasks and resources [23,33].

III. SimJava

SimJava is the fundamental event based simulation toolkit used in both CloudSim and GridSim.

IV. Cloud Analyst

Cloud Analyst is a graphical simulation tool based on Cloudsim for modeling and analyzing behavior of cloud computing environments, which supports visual modeling and simulation of large-scale applications that are deployed on cloud Infrastructures. The Cloud Analyst is built on top of CloudSim tool kit, by extending CloudSim functionality with the introduction of concepts that model Internet and Internet Application behaviors [23].

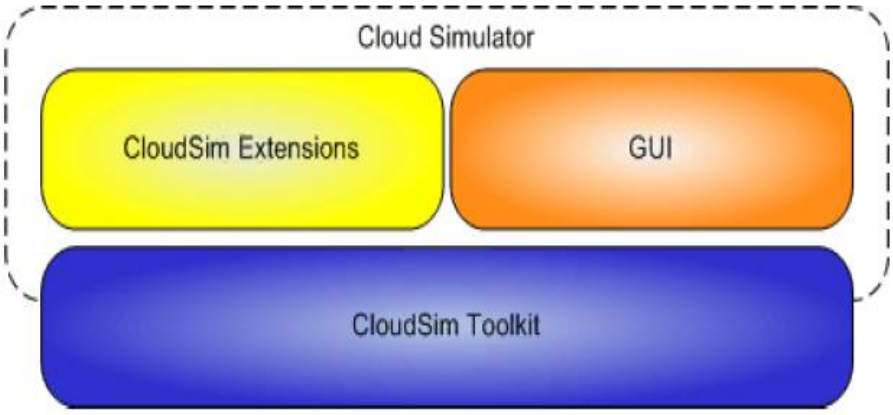


Figure 3.4: CloudAnalyst built on top of CloudSim toolkit [23].

3.7. Evaluation Approach

Based on the cloud analyst open source tool the task scheduling algorithm is evaluated in different parameters such as response time, data center processing time, data transfer cost and others. In the existing task scheduling algorithm when number of cloud users increases to access cloud resources the complexity of the task scheduling algorithm also increases and there is energy consumption because of overloaded problem. But in the propose scheduling algorithm if there is an overloaded virtual machine the task migrate to under loaded virtual machine to minimize energy consumption. Due to this we can reduce the complexity of the algorithm and increased the performance of the algorithm.

CHAPTER FOUR

4. PROPOSED SOLUTION

In this chapter, the proposed solution has presented to distribute workloads across multiple computing resources. The main objective of this chapter is to design a hybrid task scheduling algorithm to assign the users' tasks to multiple computing resources in cloud environment. The proposed algorithm is considered an incorporation of the PSO and ACO algorithms. According to the proposed Hybrid PSACO task scheduling algorithm; PSO algorithm is used to schedule the users' tasks in best fitness manner. ACO algorithm is used to determine the best path for resource allocation and shares workload for all VMs.

4.1. Mathematical Model of Task Scheduling

Let say there are n set of task or request need to be scheduled across the data center given as follow:

$$T= \{T_1, T_2 \ldots\ldots T_n\} \ldots\ldots (1)$$

And there is m set of virtual machine in data center given as:

$$V= \{V_1, V_2 \ldots\ldots Vm\} \ldots\ldots (2)$$

The current data center load when mapping a set of user task/request T to the set of virtual machines V represents as:

$$DCL= \{VL_1, VL_2 \ldots VLm\} \ldots\ldots (3)$$

Where
$$\begin{cases} VL_1 = \text{A set of tasks mapped on } V_1 \\ VL_2 = \text{A set of tasks mapped on } V_2 \\ . \\ . \\ . \\ VL_m = \text{A set of tasks mapped on } V_m \end{cases}$$

The time needed for executing all tasks t_n in one virtual machine Vi should be calculated as below:

$$T_1=\sum t_i \quad i= \{1, 2\ldots n\} \ldots\ldots (4)$$

There is more than one virtual machine and a set of tasks, all tasks can be shared to multiple server nodes dealing with in parallel, the time needed T_t represented as below:

$$Tt= \max \{ti\} \; i= \{1, 2 \ldots n\} \ldots\ldots (5)$$

The objective of this research is to minimize response time, processing time and total cost including task transfer cost and virtual machine cost when assign multiple user tasks/requests to multiple cloud resources.

Response time is calculated based on the time taken DC to receive user request, the time recorded by DC and the delay time the request to reach in DC.

$$\mathrm{Rt} = \mathrm{Ft} - \mathrm{At} + \mathrm{Dt} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

Where
- Rt= Response time
- Ft= Finish time
- At=Arrival Time
- Dt=Delay Time

DC processing time is calculated based on user request per hour and bandwidth allocation

$$\mathrm{PT} = \frac{R/hr.}{BW} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7)$$

Where
- PT = processing time,
- R/hr.= request per hour and
- BW = bandwidth allocation

Task transfer cost is a cost where transfer or migrate a task from overloaded virtual machine to underloaded virtual machine.

$$\mathrm{TTC} = \mathrm{Cost} + \mathrm{BWcost} * \mathrm{Tsize} \quad \ldots\ldots\ldots\ldots\ldots\ldots (8)$$

Where
- TTC = task transfer cost,
- BWcost =cost per bandwidth,
- Tsize is total task size
- Cost=0

Virtual machine cost is the cost of cloud resource based on memory cost, storage cost and processor cost.

$$\mathrm{VMC} = \mathrm{Mcost} + \mathrm{Scost} + \mathrm{Pcost} \quad \ldots\ldots\ldots\ldots\ldots\ldots (9)$$

Where, VMC is virtual machine cost, Mcost is memory cost, Scost is storage cost and Pcost is processor cost.

Based on the above 8&9 equation the objective function of total cost is represent as below:

$$F(TC) = \sum_{i=1}^{n} TTCi + \sum_{j=1}^{m} VMCj \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (10)$$

Where

F (TC) = Total cost

TTC_i= transfer cost of task, i= {1, 2 …. n}

VMC_j= virtual machine cost j= {1, 2 ….m}

4.2. Proposed algorithm

In this research, we proposed a hybrid task scheduling algorithm that takes advantages of PSO and ACO algorithms. The convergence speed of PSO algorithm is fast in the initial stage, but the local searching ability will decrease in the later due to the lack of population diversity. In order to improve the above problem ACO is used. ACO has better optimization ability and the early convergence speed is slow due to the lack of pheromone in an initial stage. ACO method does not use controlled global schedulers to finish the allocation of resources within a given time. Whereas in PSO is best for its low computational cost and high throughput during run time. It helps to find the optimal solution efficiently [24].

In the proposed algorithm, get the user request from data center broker and then prioritize the incoming request based on fitness value. Proposed algorithm evaluates the fitness value of each particle (task) to get best particle position and put in swarm/information list. Finally, select suitable VMs for the incoming request based on resource and task information and to get the optimal solution of the task scheduling problem.

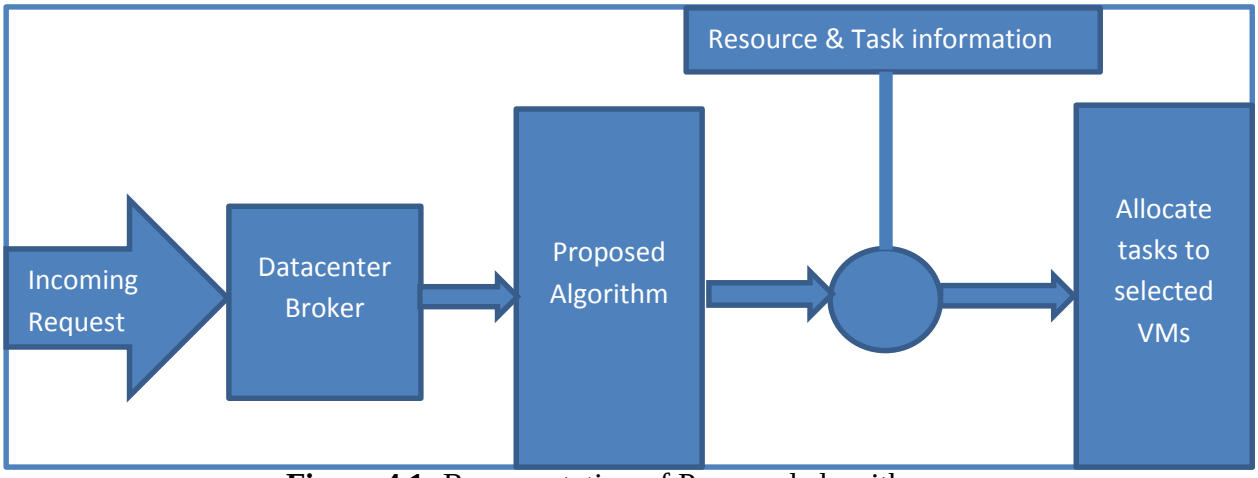


Figure 4.1: Representation of Proposed algorithm

4.3. Fitness Calculation

The PSO algorithm is a population-based search algorithm based on the simulation of the social behavior of birds within the flock and fish school proposed by Kennedy and Eberhart in 1995 [25,41].

During N-Dimension population search space the i^{th} particle position (X_i) and velocity (V_i) of the swarm represented as:

$$X_i = (X_{i1}, X_{i2} \ldots\ldots X_{iN})$$

$$V_i = (V_{i1}, V_{i2} \ldots\ldots V_{iN})$$

The Particle updates its position (X_i) and velocity (V_i) using the following equation:

$$V_i^{(k+1)} = wV_i^{(k)} + c1\,r1 * (pb_i - X_i^{(k)}) + c2r2 * (gb - X_i^{(k)}) \quad \ldots\ldots(11)$$

$$X_i^{(k+1)} = X_i^{(k)} + V_i^{(k+1)} \quad \ldots\ldots(12)$$

Where

- $V_i^{(k)}$ = velocity of particle i at iteration k.
- $X_i^{(k)}$ = position of particle i at iteration k.
- w= inertia weight
- $V_i^{(k+1)}$ = velocity of particle i at iteration k+1.
- $X_i^{(k+1)}$ = position of particle i at iteration k+1.
- r1, r2 = random number between (0, 1)
- c1, c2 = acceleration coefficient.
- pbi =best position of particle i
- gb= position of best particle in a population

4.4. Flow chart of proposed algorithm

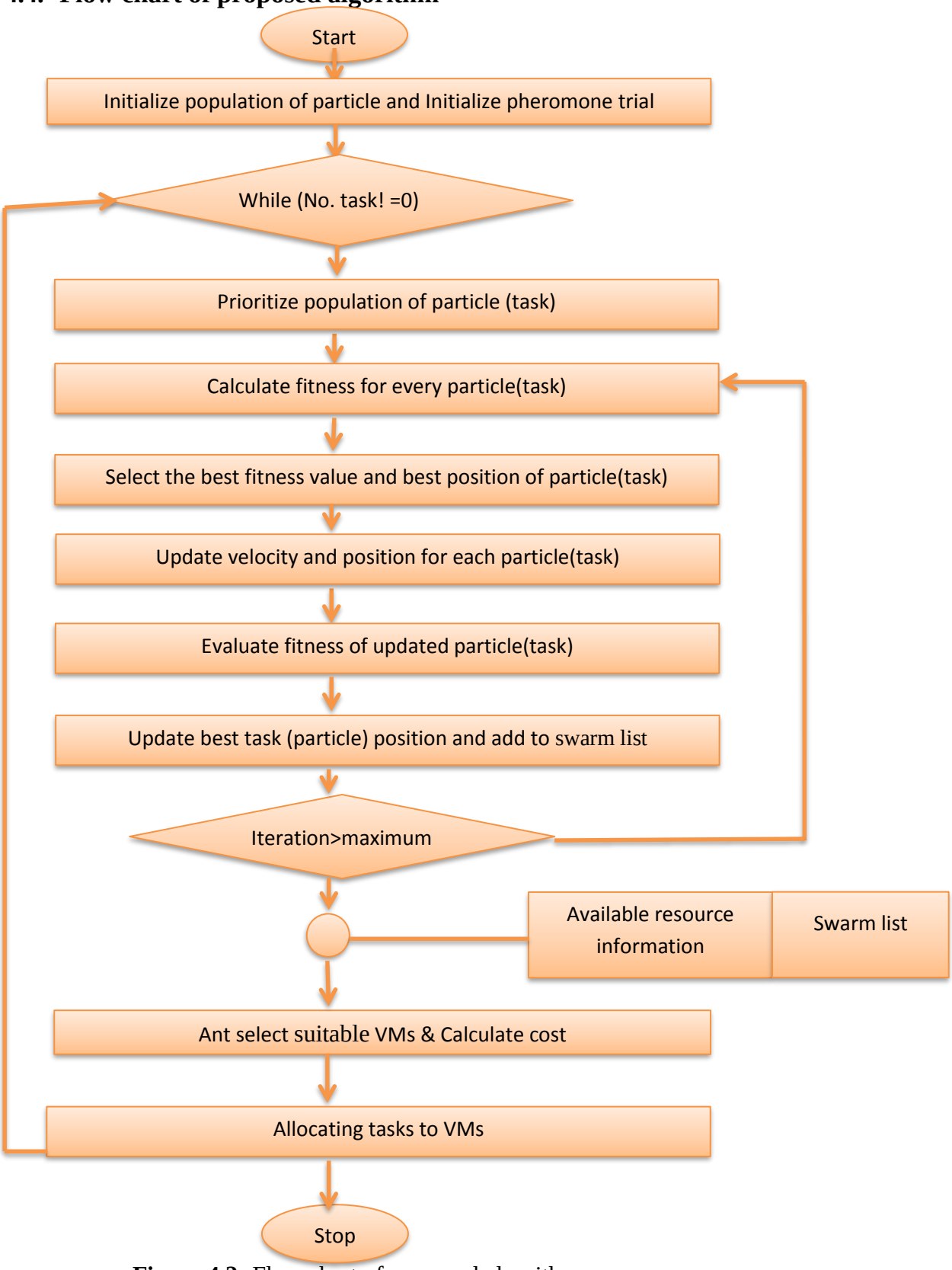


Figure 4.2: Flow chart of proposed algorithm

4.5. Proposed algorithm Programming steps

1. Initialize population and pheromone with random solution

2. While (No. task! =0) do

3. Prioritize population

4. Calculate the fitness for each tasks (particles) using the object function

5. Compare the current particle fitness value with other particle fitness value and find local best value (pbest).

6. Compare the pbest of population and fitness value to get global best (Gbest).

7. Update each particle velocity and position vector using equation 11 & 12

8. Evaluate the fitness of updated particle (task)

9. Update best particle position (Gbest) and put in swarm list

10. If the number of iteration reached maximum go to step 11 otherwise repeat from step 4.

11. Select suitable VMs based on resource and task information and calculate VMs cost using Equ. (9)

12. If VM is available and least loaded go to step 13 otherwise transfer task to other least loaded VMs and calculate transfer cost using Equ. (8).

13. Allocate task to VMs and calculate Response time using Equ. (6), processing time using Equ. (7) and overall cost using Equ. (10)

14. If the number of ant reached maximum stop, otherwise repeat from step 2.

CHAPTER FIVE

5. EXPERIMENTAL DESIGN AND IMPLEMENTATION

5.1. Introduction

In this chapter, the proposed algorithm has been implemented using Cloud analyst simulation tool and evaluate the simulation result based on different number of independent tasks and different number of VMs and compare the performance relative to the standard PSO, ACO and ABC in terms of cost, response time and data center processing time. Java language is used for development and implementation of proposed algorithm because the cloud analyst simulation tool is developed by java language and easy to learn.

5.2. Cloud analyst

Cloud Analyst is a GUI simulation tool that developed on cloudsim architecture for modeling, analyzing and experimenting behavior of cloud computing environments, which supports visual modeling and simulation of different applications that are deployed on cloud Infrastructures.

The cloud analyst allows setting location of users that are generating the application and the location of the data centers. In this various configuration parameters can be set like number of users, number of request generated per user per hour, number of virtual machines, number of processors, amount of storage, network bandwidth and other necessary parameters. Based on the parameters the tool computes the simulation result and shows them in GUI form. The result includes response time, processing time, cost etc. [26].

The main features of Cloud Analyst are as below [28]: -

- Easy to use Graphical User Interface (GUI)
- Capability to define a simulation with a high degree of configurability and flexibility.
- Repeatability of experiments.
- Graphical output.
- Use of consolidated technology and ease of extension

The basic components of cloud analyst are as follow [27]:

- **Region** -In the CloudAnalyst the world is divided in to 6 ‘Regions’ that coincide with the 6 main continents in the World. The other main entities such as User Bases and Data Centers belong to one of these regions [29].
- **GUI Packages-** It is mainly responsible for the graphical user Interface.
- **User Base -** A User Base models a group of users that is considered as a single unit in the simulation and generates traffic for the simulation**.**
- **Internet**- It models the Internet traffic routing around the globe by introducing transmission and data transfer delays.
- **DataCenterController**-It generally controls Data center activities.
- **VmLoadBalancer**- the Data center controller generally uses a VmLoadBalancer to determine which VM should be assigned the next Cloudlet for processing and models load balancing policies
- **Simulation**-This is used for creating and executing the request.
- **CloudAppServiceBroker** – This component model service brokers that handle traffic routing between user bases and data centers.

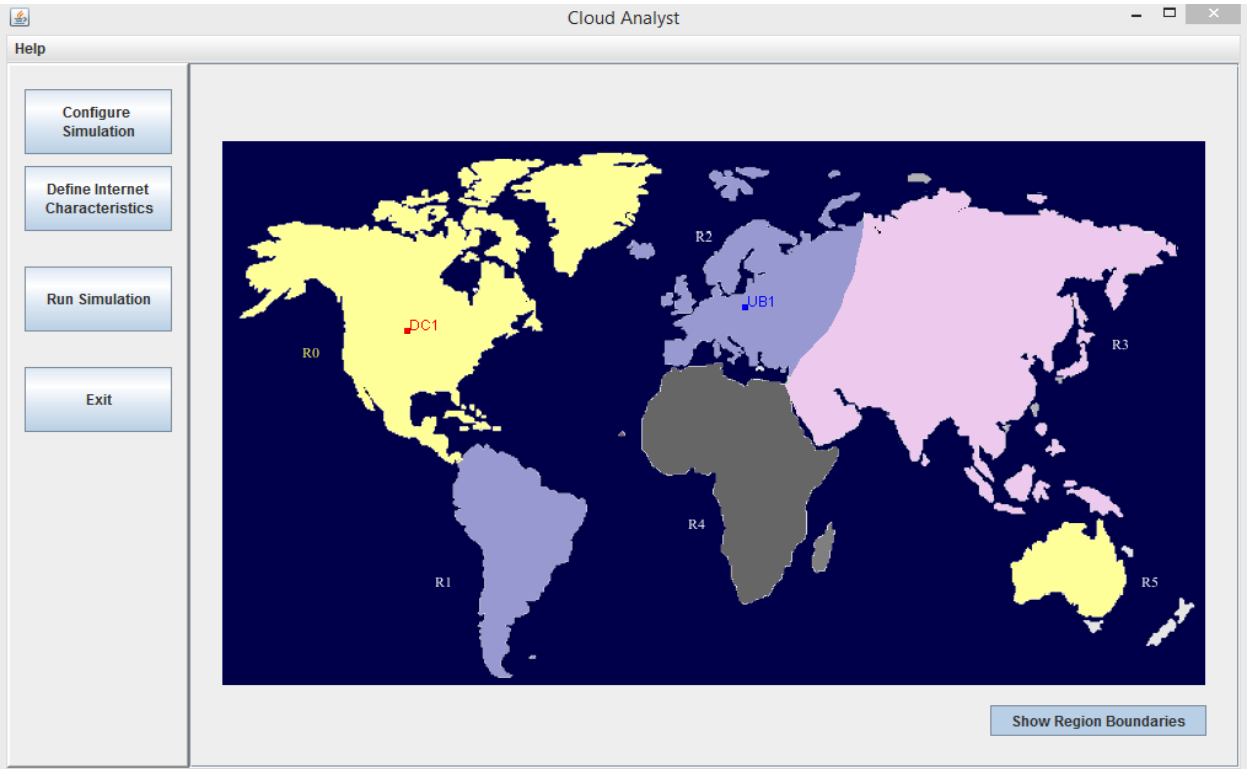


Figure 5.1: CloudAnalyst GUI

In cloud analyst GUI, the world divided into six regions (R1, R2, R3, R4, R5, & R6) which are located on the right side of simulation panel. The main control panel option in the cloud analyst main screen located on left side.

The main control panel options are:

- **Configure Simulation**: this option takes us to the Configure Simulation Screen.
- **Define Internet Characteristics**: which takes us to the Internet Characteristics Screen
- **Run Simulation:** used to run the simulation
- **Exit**

5.2.1. Configure Simulation Screen

Configure Simulation screen has main configuration; data center configuration and advanced tabs. To Configure Simulation screen, we carried out the following steps:

- Define the duration of the simulation in terms of minutes, hours and days.
- Define UserBases characteristics
- Define Datacenter characteristics
- Allocate Virtual Machines for the application in each Data Centers
- Define the scheduling & load balancing policy across VMs in a data center.
- Adjust the Internet Characteristics (bandwidth matrices and network latency).

Main configuration

In main configuration tab simulation duration, service broker policy, user bases characteristics and application deployment configuration must be defined.

Simulation duration define in terms of minutes, hours and days. User bases characteristics has the following configurable fields: Name, Region, Requests per user per hour, Data size per request, Peak hours Start, Peak Hours End, Average users during peak hours, Average users during off-peak hours. Application Deployment Configuration has Data Center, Number of VMs, Image Size, Memory and BW fields. Service Broker Policy has two drop down list: -

Closest data center – The data center with the least network latency from a particular userbase is sent all the requests from that user base.

Optimize response time – This policy attempts to balance the load between data centers when one data center gets over loaded.

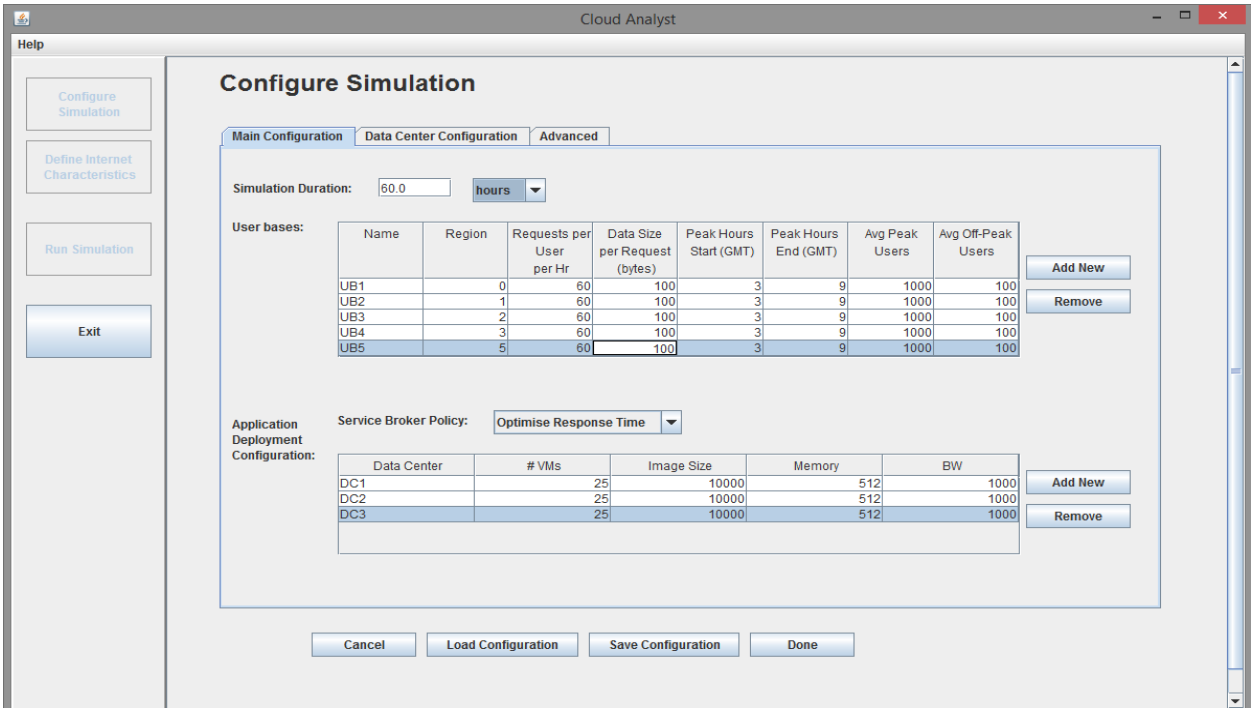


Figure 5.2: Main configuration screen

Data center configuration

In the data center configuration tab, the values of each data center parameters and the details of each physical hardware of each data center must be configured.

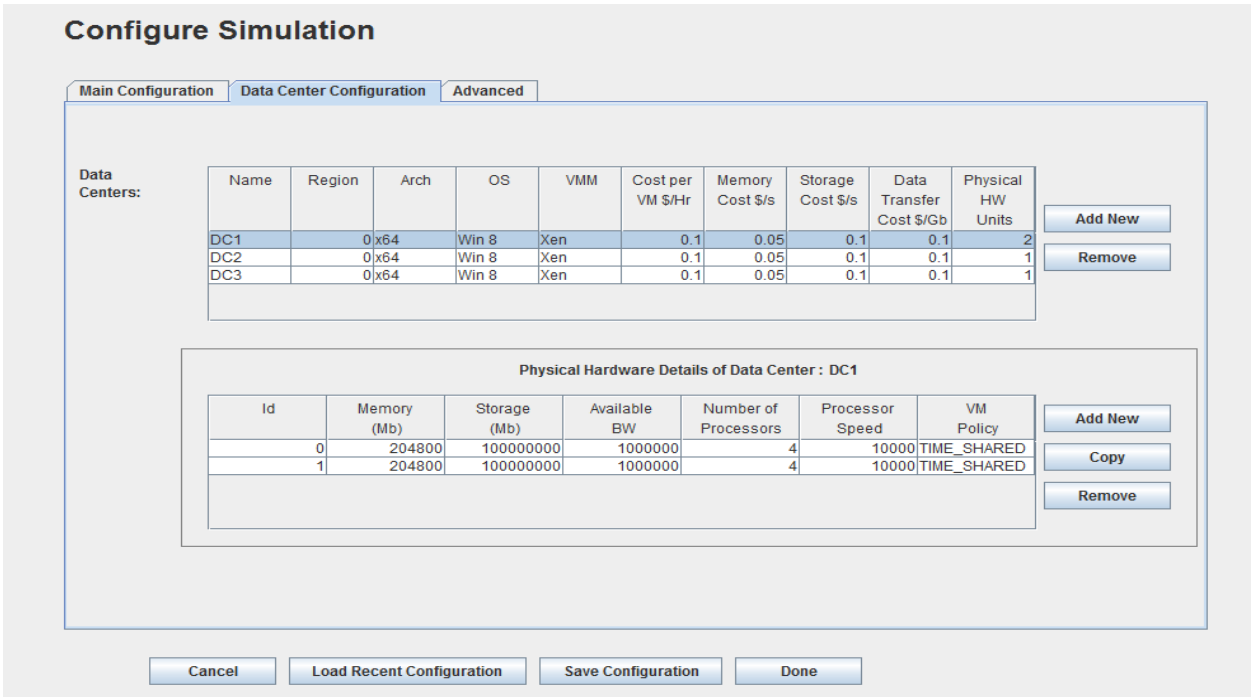


Figure 5.3: Data center configuration screen

Advanced tab

The advanced tab contains an important parameter that apply to the entire simulation.

User Grouping Factor (in UB) – This parameter tells the simulator how many users should be treated as a single bundle for traffic generation. The number given here will be used as the number of requests represented by a single Internet Cloudlet.

Request Grouping Factor (in DC) – This parameter tells the simulator how many requests should be treated as a single unit for processing. i.e. these many requests are bundled together and assigned to a single VM as a unit.

Executable instruction length (in bytes) – This is the main parameter that affects the execution length of a request. This is the same Gridlet Length parameter used in GridSim.

Scheduling and Load balancing policy – the scheduling and load balancing policy used by all data centers in allocating requests to virtual machines. Available policies are: Hybrid PSACO, Ant colony, Particle Swarm and Artificial bee colony (ABC)

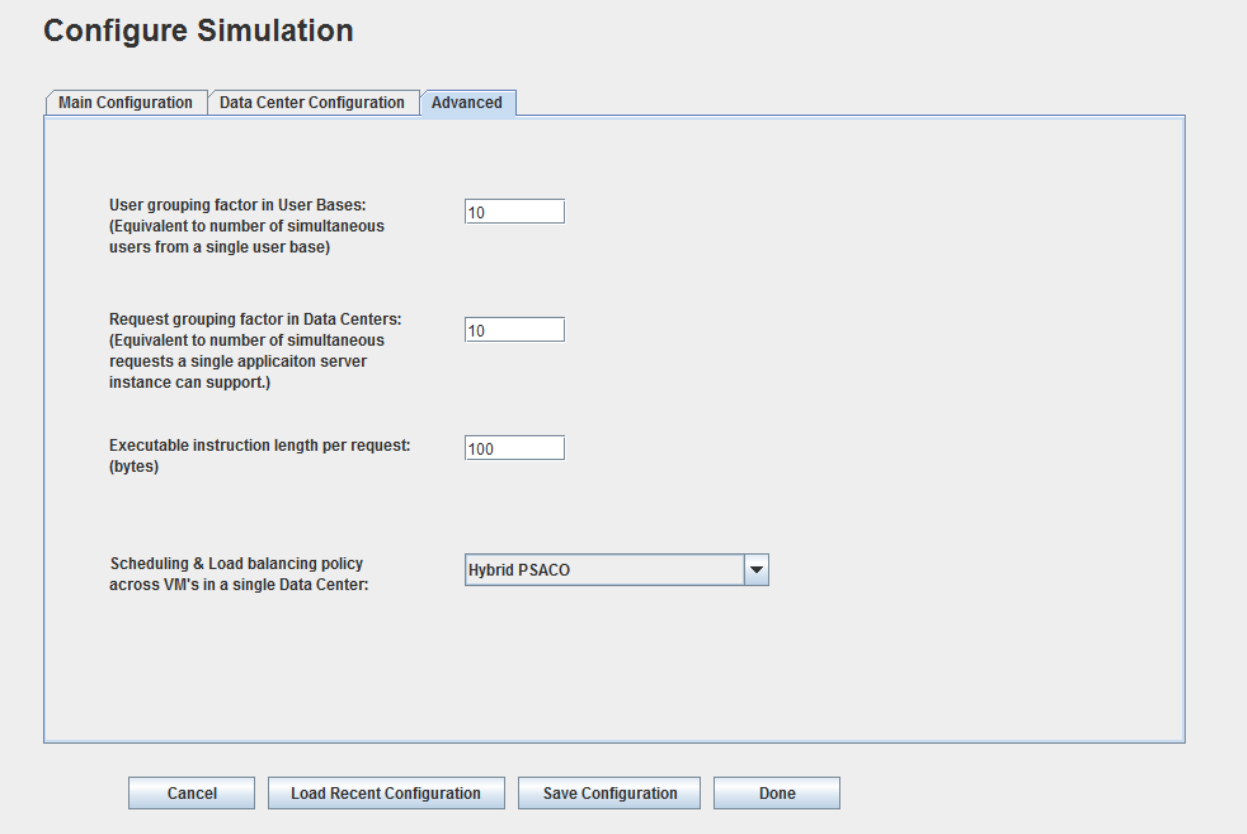


Figure 5.4: Advanced tab in configure simulation

5.2.2. Define Internet Characteristics

Internet Characteristics define in terms of bandwidth matrices and network latency.

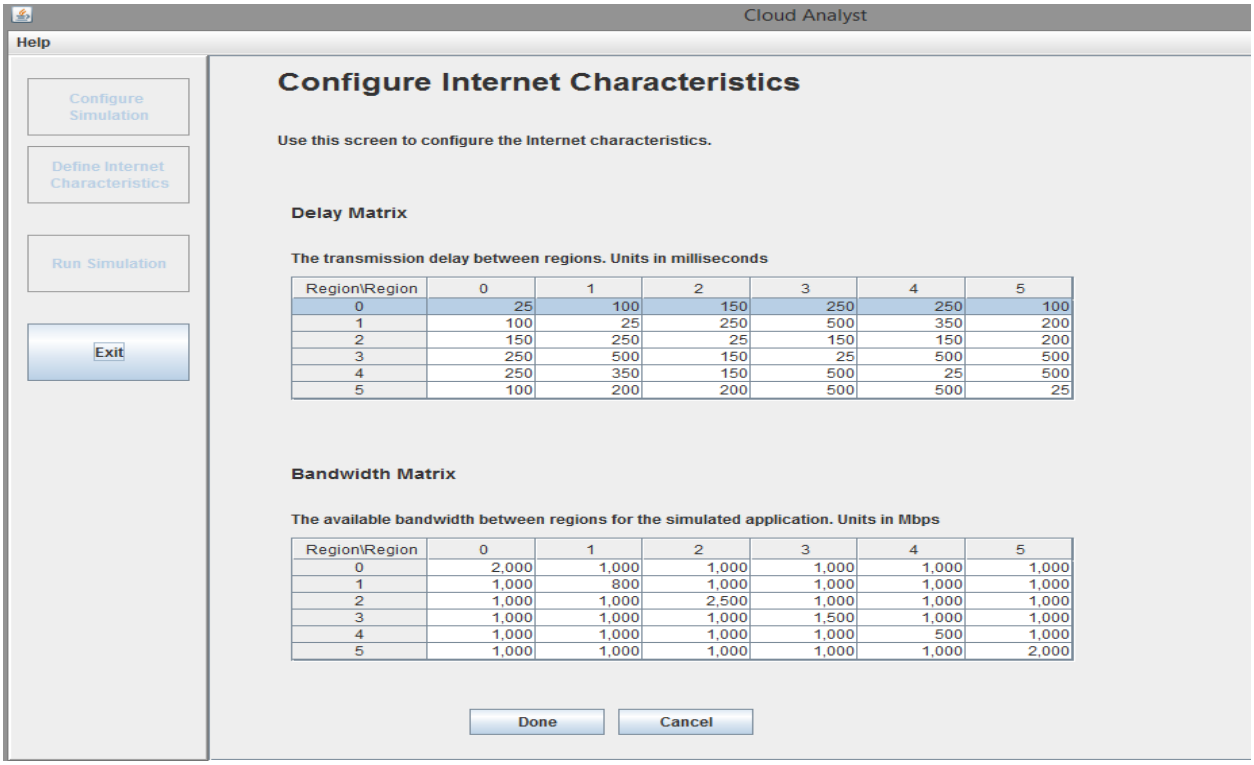


Figure 5.5: Internet Characteristics Screen

5.2.3. Run Simulation

After defining configure simulation and internet characteristics the next step is running simulation to get simulation result.

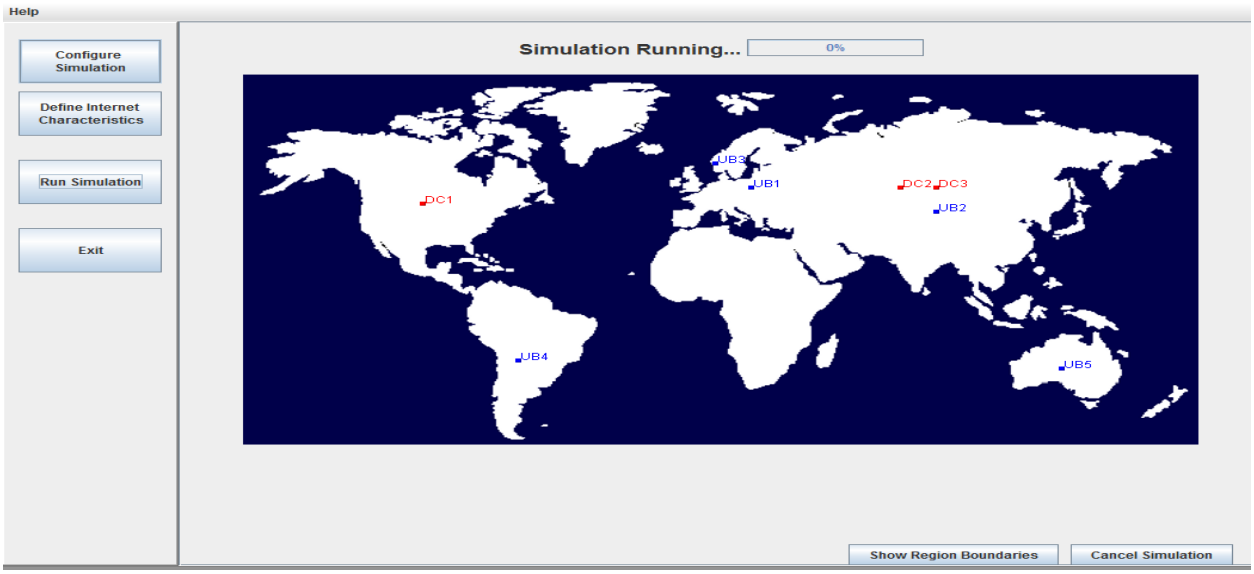


Figure 5.6: Running simulation Screen

Table 5.1: Parameters used in Experiments

Parameter	Value
Arch	X64
Operating System	Win 8
Virtual Machine Management VMM	Xen
VM Image Size	10000
VM Memory (MB)	512-1024
VM Bandwidth	10000
DC Memory/Machine(MB)	2048
DC Storage/Machine (MB)	100000
DC Available BW/Machine	1000000
DC processors Speed(MIPS)	10000
VM policy	TIME_SHARED
Cost per VM /hour	$0.10
Data transfer cost /1Gb of (from/to Internet)	$0.10
Storage cost/s	$0.10
Memory cost/s	$0.05

5.3. Experiment and Result

In our experiment, we used a single data center (DC) with 50 VMs deployed in a single location, region two (R2), a multiple data centers (DCs) with different number of virtual machine (VMs) and physical processors deployed in six different locations (America, Middle East, Europe, Asia, Africa and Oceania/Australia). Each virtual machines running on physical processors with capable of 10000MIPS processors speed. The proposed algorithm evaluates in the experiment based on response time, datacenter processing time and cost. We compare the hybrid algorithm with ACO, ABC and PSO algorithms.

Experiment 1:

In this experiment, the application deployed in one data center(DC) with 50 virtual machine (VMs) in a single environment and six different user bases (UB1, UB2, UB3, UB4, UB5, UB6) to test the scheduling algorithms. Each user base represents thousands of internet users which located in region 2 (R2). The data center physical hardware has four processors each processor has 10000MIPS speed. Users are grouped by a factor of 1000 in user bases, and requests are grouped by a factor of 100 in data center. Each user request requires 500 bytes' instructions to be executed. The simulation duration took 60 minutes. For evaluation, we used the response time and data center processing time metrics to compare the proposed algorithm with other current meta-heuristics algorithms.

Table 5.2: Experiment 1 User base Configurations

Name	Region	Requests per user per Hr.	Data Size per request(Bytes)	Peak Hours Start(GMT)	Peak Hours End(GMT)	Avg. Peak Users	Avg. Off-peak Users
UB1	2	150	100	3	9	72432	7242
UB2	2	150	100	3	9	14697	1469
UB3	2	150	100	3	9	65963	6596
UB4	2	150	100	3	9	193807	19380
UB5	2	150	100	3	9	38837	3883
UB6	2	150	100	3	9	2818	281

Table 5.3: Experiment 1 Application Deployment configuration

Data Center	#VM	Image Size	Memory	BW
DC1	50	10000	1024	1000

Table 5.4: Experiment 1 Data center physical hardware configuration

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	10000	TIME_SHARED
1	204800	100000000	1000000	4	10000	TIME_SHARED

Result

From this experiment, we obtain the following result in table 5.5:

Table 5.5: Experiment 1 result

Algorithms		Average (ms.)	Minimum(ms)	Maximum(ms)
ABC	Response time	411.23	252.51	578.56
	Data Center processing time	85.31	1.02	250.28
ACO	Response time	322.67	251.37	400.05
	Data Center processing time	18.87	0.69	35.51
PSO	Response time	322.70	238.81	402.30
	Data Center processing time	18.80	0.69	63.24
PSACO	Response time	321.89	251.37	400.05
	Data Center processing time	18.19	0.69	32.01

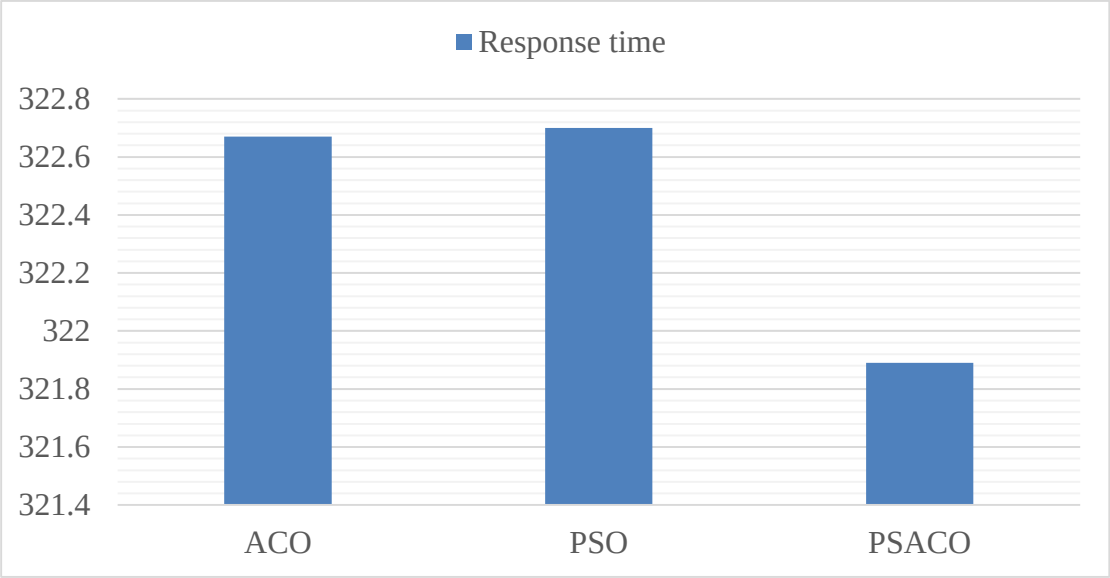


Figure 5.7. Comparison based on Response time in single location

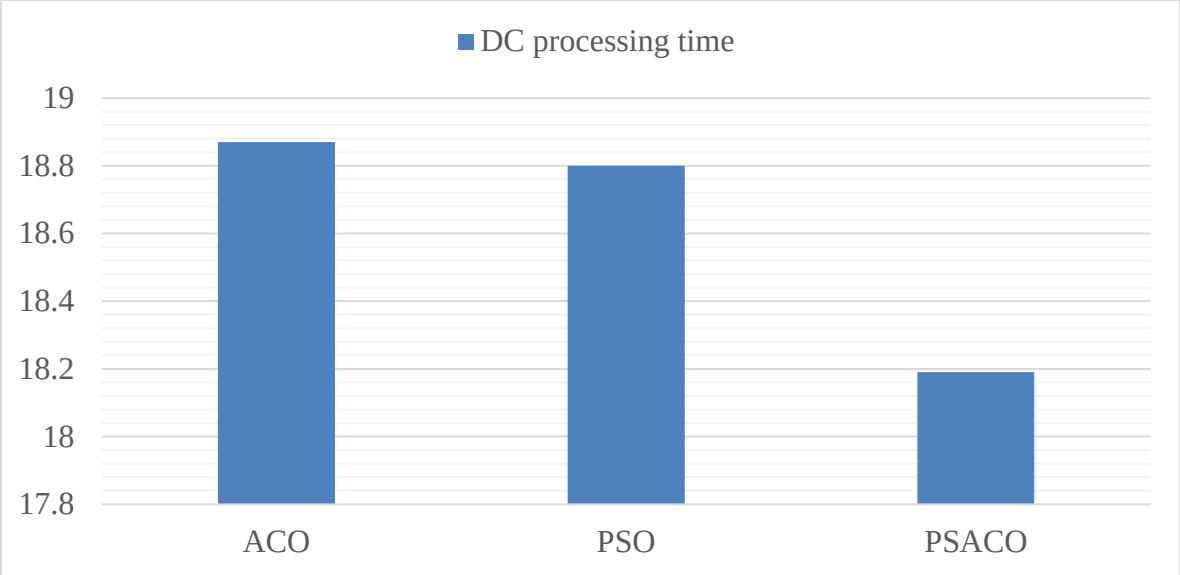


Figure 5.8: Comparison based on DC processing time in single location

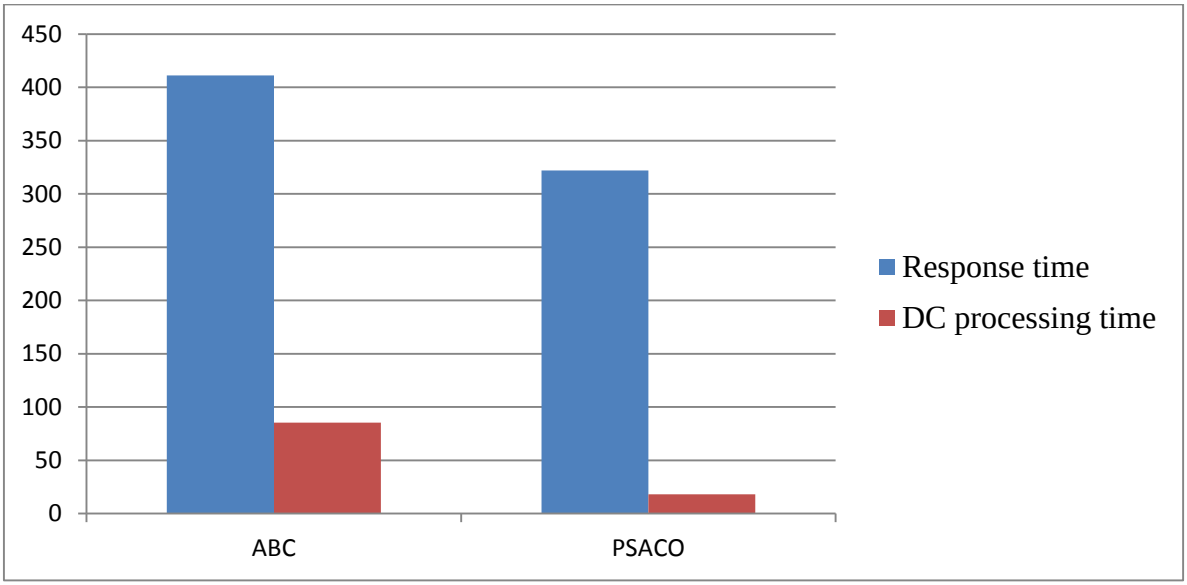


Figure 5.9: Comparison based on DC processing time and Response time in single location

Based on the above tables and figures the propose hybrid has a better response time and data center processing time than the existing algorithms. The average response time of PSACO is 321.89 (ms) and the average processing time is 18.19 (ms). This result better than ABC algorithms, the average response time of ABC is 411.23(ms) and the average processing time is 85.31 (ms). The proposed algorithm minimizes the average response time by 89.34(ms) and the average processing time by 67.12 (ms) than ABC algorithm.

Experiment 2:

In this experiment, the application deployed in six different locations (R0, R1, R2, R3, R4 &R5) and each location has user bases (UB5, UB4, UB6, UB2, UB3, UB1) respectively). The application deployed in two data centers (DCs) DC1 and DC2. DC1 allocated in America (R0) and DC2 allocated in Europe (R3 with 25 virtual machine (VMs) for each DCs. Users are grouped by a factor of 1000 in user bases, and requests are grouped by a factor of 100 in DC. Each user request requires 500 bytes' instructions to be executed. The simulation duration took 30 minutes. The data center physical hardware configuration file is as experiment 1 table 5.4.

Table 5.6: Experiment 2 User base Configurations

Name	Region	Requests per user per Hr.	Data Size per request(Bytes)	Peak Hours Start(GMT)	Peak Hours End(GMT)	Avg. Peak Users	Avg. Off-peak Users
UB1	R5	150	100	3	9	72432	7242
UB2	R3	150	100	3	9	14697	1469
UB3	R4	150	100	3	9	65963	6596
UB4	R1	150	100	3	9	193807	19380
UB5	R0	150	100	3	9	38837	3883
UB6	R2	150	100	3	9	2818	281

Table 5.7: Experiment 2 Application Deployment configuration

Data Center	#VM	Image Size	Memory	BW
DC1	25	10000	1024	1000
DC2	25	10000	1024	1000

Result

From this experiment, we obtain the following result in table 5.8:

Table 5.8: Experiment 2 result

Algorithms		Average (ms.)	Minimum(ms)	Maximum(ms)
ABC	Response time	285.09	47.49	665.54
	Data Center processing time	47.50	0.56	127.57
ACO	Response time	242.34	45.30	616.46
	Data Center processing time	10.11	0.36	45.70
PSO	Response time	242.31	44.92	622.80
	Data Center processing time	10.14	0.37	29.84
PSACO	Response time	241.46	44.92	616.46
	Data Center processing time	9.44	0.37	15.58

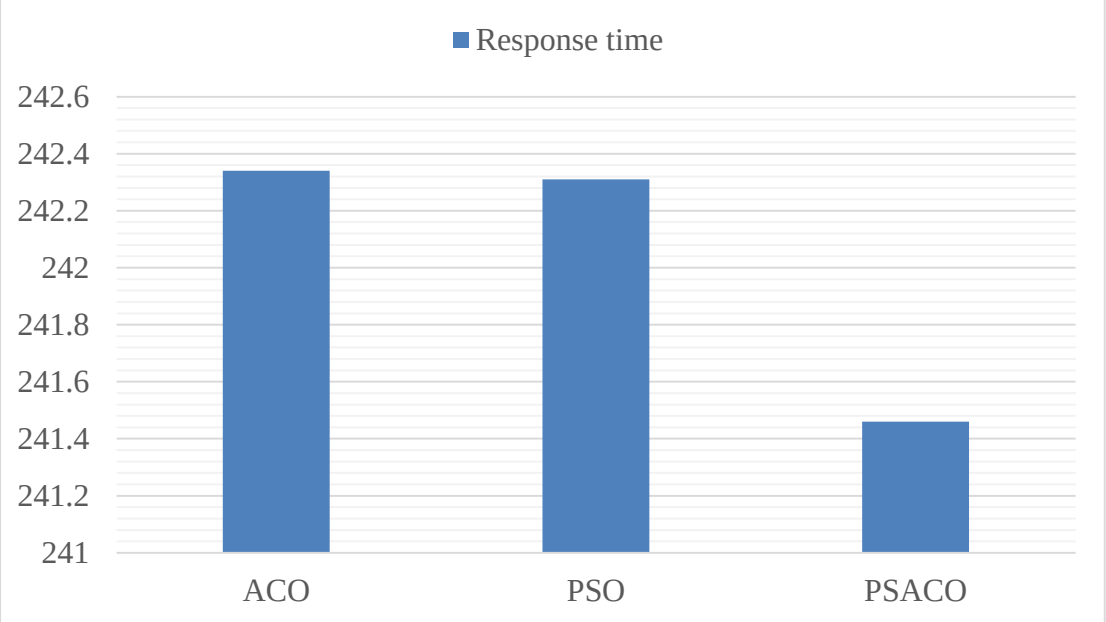


Figure 5.10 Comparison of algorithms based on Response time in six different location

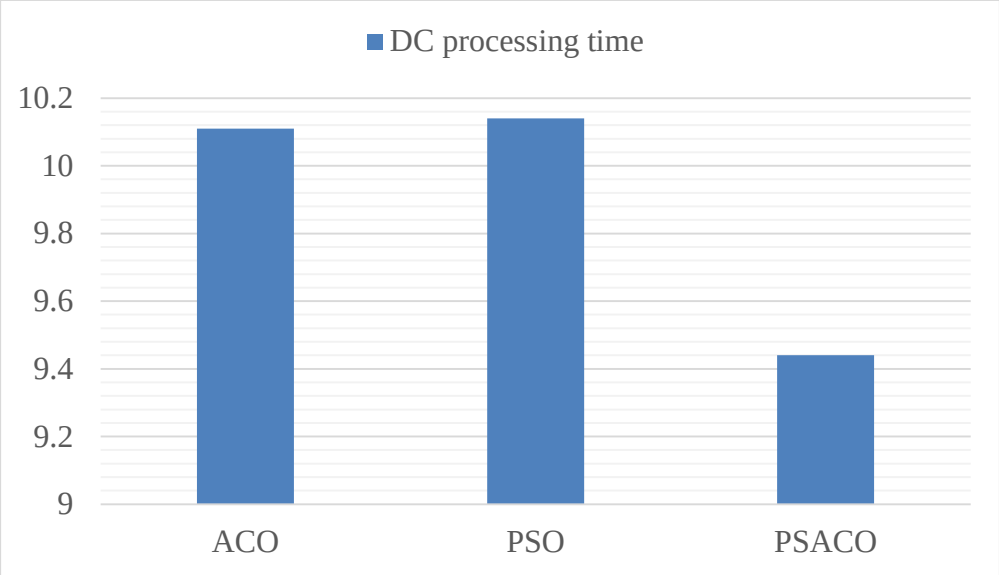


Figure 5.11: Comparison algorithms based on DC processing time in six different location

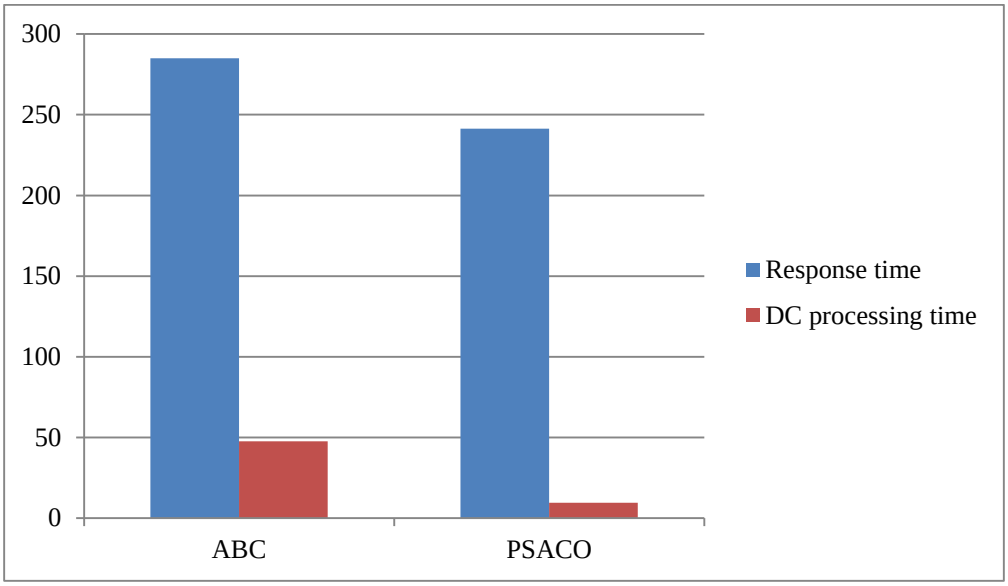


Figure 5.12: Comparison based on DC processing time and Response time in six different location

Based on the experiment 5.2, the above tables and figures shows that the propose hybrid has a better response time and processing time than the existing algorithms in six different regions. The average response time of PSACO is 241.46 (ms) and the average processing time is 9.44 (ms). This result better than ABC algorithms, the average response time of ABC is 285.09 (ms) and the average processing time is 47.50 (ms). The proposed algorithm minimizes the average response time by 43.63(ms) and the average processing time by 38.06 (ms) than ABC algorithm.

Experiment 3:

In this experiment, the application deployed in six different heterogonous locations (R0, R1, R2, R3, R4 &R5) with 12 different user bases (12UB). The application deployed in 5 data centers (DCs) with different number of VMs, DC1-15 VMs, DC2-25 VMs, DC3-50 VMs, DC4-75 VMs and DC5-100 VMs. DC1 allocated R0, DC2 allocated R1, DC3 allocated in R2, DC4 allocated in R3 and DC5 allocated in R4. The data center physical hardware configuration file is as experiment 1 table 5.4.

Table 5.9: Experiment 3 User base Configurations

Name	Region	Requests per user per Hr.	Data Size Per request(Bytes)	Peak Hours Start(GMT)	Peak Hr. End (GMT)	Avg. Peak Users	Avg.Off-peak Users
UB1	R0	150	100	3	9	72432	7242
UB2	R1	150	100	3	9	14697	1469
UB3	R2	150	100	3	9	65963	6596
UB4	R3	150	100	3	9	193807	19380
UB5	R4	150	100	3	9	38837	3883
UB6	R5	150	100	3	9	2818	281
UB7	R2	150	100	3	9	65963	6596
UB8	R5	150	100	3	9	2818	281
UB9	R0	150	100	3	9	72432	7242
UB10	R4	150	100	3	9	38837	3883
UB11	R1	150	100	3	9	14697	1469
UB12	R3	150	100	3	9	193807	19380

Table 5.10: Experiment 3 Application Deployment configuration

Data Center	#VM	Image Size	Memory	BW
DC1	15	10000	1024	1000
DC2	25	10000	1024	1000
DC2	50	10000	1024	1000
DC3	75	10000	1024	1000
DC4	100	10000	1024	1000

Result

From this experiment, we obtain the following result in table 5.11:

Table 5.11: Experiment 3 result

Algorithms		Average (ms.)	Minimum(ms)	Maximum(ms)
ABC	Response time	192.68	41.40	604.76
	Data Center processing time	91.71	0.58	466.42
ACO	Response time	74.52	41.62	248.63
	Data Center processing time	21.86	0.21	0.21
PSO	Response time	74.72	41.40	248.63
	Data Center processing time	22.07	0.21	81.45
PSACO	Response time	73.61	41.62	248.63
	Data Center processing time	21.16	0.21	62.72

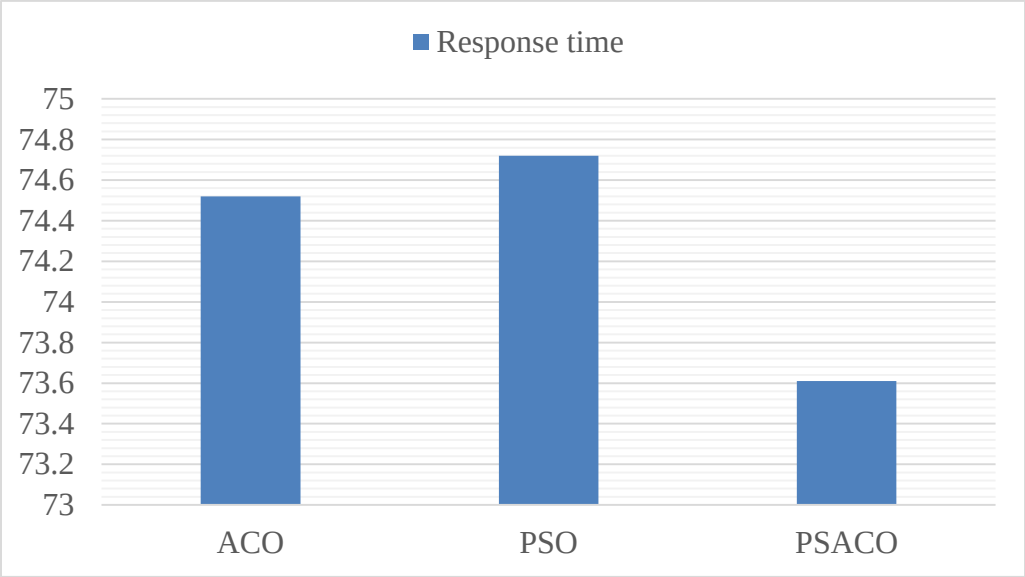


Figure 5.13 Comparison of algorithms based on Response time in 12 different User Bases

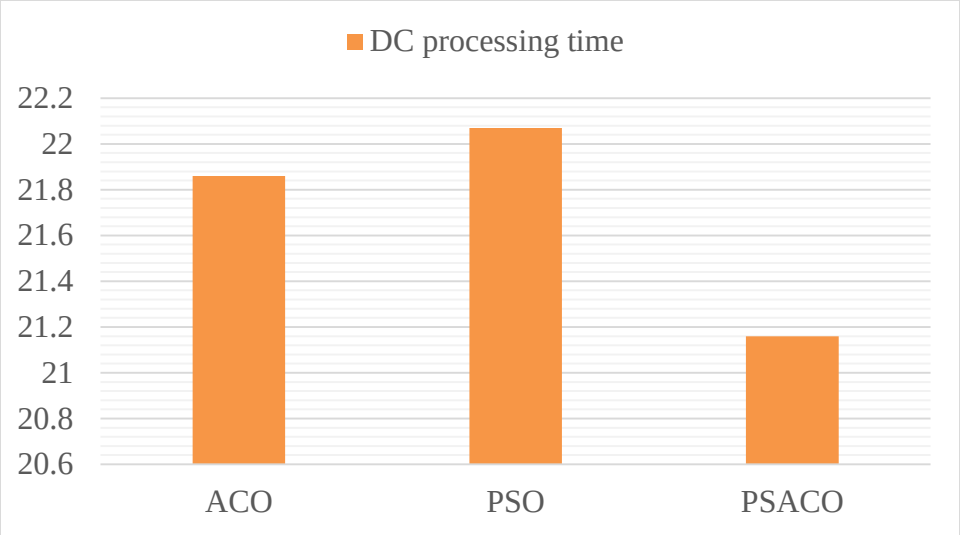


Figure 5.14: Comparison algorithms based on DC processing time in 12 different User Bases

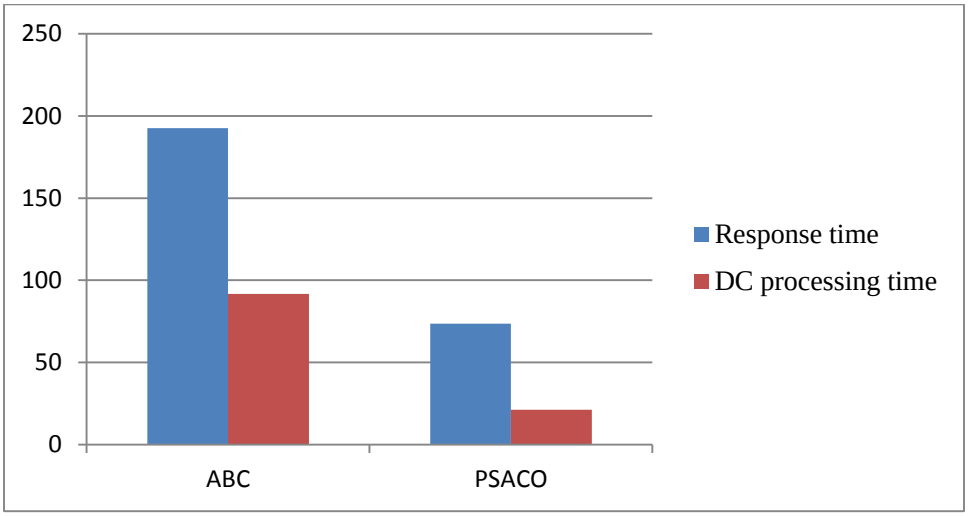


Figure 5.15: Comparison based on DC processing time and Response time in 12 different user bases

In the experiment 5.3, the above tables and figures shows that the propose hybrid has a better response time and processing time than the existing ABC, ACO and PSO algorithms in 12 different user bases. The average response time of PSACO algorithm is 73.61 (ms) and the average processing time is 21.16 (ms). This result better than ABC algorithms, the average response time of ABC is 192.68 (ms) and the average processing time is 91.71 (ms). The proposed algorithm minimizes the average response time by 119.07(ms) and the average processing time by 70.55 (ms) than ABC algorithm.

Experiment 4:

In this experiment, the number of users become large and the application deployed in six different heterogonous locations (R0, R1, R2, R3, R4 &R5) with 20 different user bases (20UB). The application deployed in 5 data centers (DCs) with different number of VMs, DC1-15 VMs, DC2-25 VMs, DC3-50 VMs, DC4-75 VMs and DC5-100 VMs. DC1 allocated R3, DC2 allocated R4, DC3 allocated in R0, DC5 allocated in R3 and DC3 allocated in R4. The application deployment configuration and data center physical hardware configuration file is as experiment 4 table 5.10 and experiment 3 table 5.10.

Table 5.12: User base Configurations

Name	Region	Requests per user per Hr.	Data Size per request(Bytes)	Peak Hr. Start(GMT)	Peak Hr. End(GMT)	Avg. Peak Users	Avg. Off-peak Users
UB1	R0	150	100	3	9	72432	7242
UB2	R1	150	100	3	9	14697	1469
UB3	R2	150	100	3	9	65963	6596
UB4	R3	150	100	3	9	193807	19380
UB5	R4	150	100	3	9	38837	3883
UB6	R5	150	100	3	9	2818	281
UB7	R2	150	100	3	9	65963	6596
UB8	R5	150	100	3	9	2818	281
UB9	R0	150	100	3	9	72432	7242
UB10	R4	150	100	3	9	38837	3883
UB11	R1	150	100	3	9	14697	1469
UB12	R3	150	100	3	9	193807	19380
UB13	R2	150	100	3	9	65963	6596
UB14	R4	150	100	3	9	38837	3883
UB15	R5	150	100	3	9	2818	281
UB16	R3	150	100	3	9	193807	19380
UB17	R1	150	100	3	9	14697	1469
UB18	R5	150	100	3	9	2818	281
UB19	R2	150	100	3	9	65963	6596
UB20	R3	150	100	3	9	193807	19380

Result

From this experiment, we obtain the following result in table 5.13:

Table 5.13: Experiment 4 result

Algorithms		Average (ms.)	Minimum(ms)	Maximum(ms)
ABC	Response time	127.71	40.40	464.88
	Data Center processing time	22.03	0.16	165.57
ACO	Response time	109.76	40.38	374.11
	Data Center processing time	6.02	0.16	21.52
PSO	Response time	170.36	42.57	7621.89
	Data Center processing time	64.03	0.58	7573.07
PSACO	Response time	109.51	40.40	414.25
	Data Center processing time	5.66	0.15	21.54

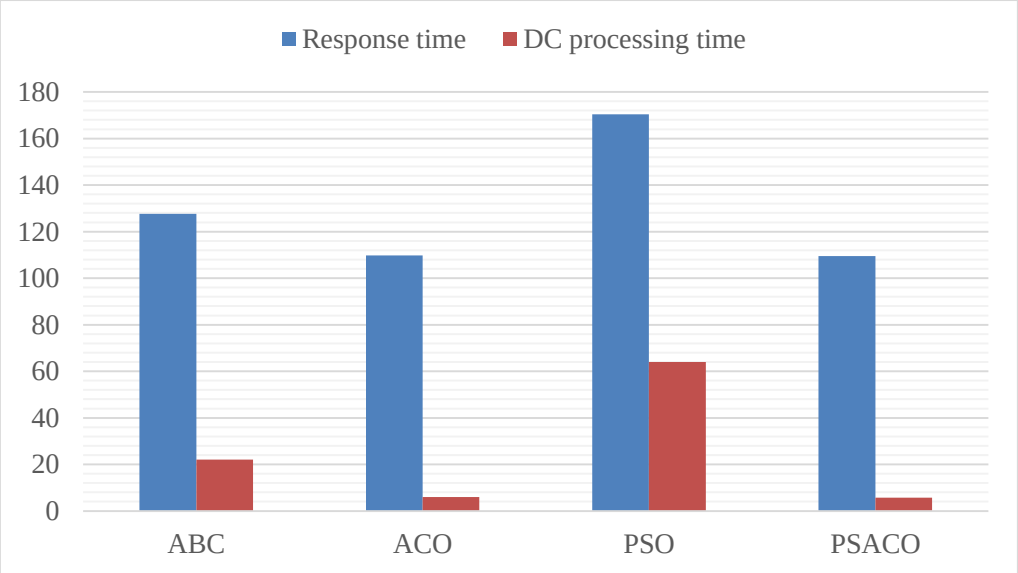


Figure 5.16 Comparison of Proposed Algorithm with 3 algorithms based on DC processing time and Response time in 20 different user bases

In the experiment 5.4, the above tables and figures shows that the propose hybrid algorithm has a better response time and processing time than the existing ABC, ACO and PSO algorithms in 20 different user bases. The average response time of PSACO is 109.51 (ms) and the average

processing time is 5.66 (ms). This result better than ABC, ACO and PSO algorithms, the average response time of ABC is 127.71 (ms) and the average processing time is 22.03 (ms). The average response time of ACO is 109.76 (ms), PSO is 170.36 and the average processing time of ACO is 6.02 (ms), PSO is 64.03. The proposed algorithm minimizes the average response time by 18.2(ms) ,0.25 (ms), 60.85(ms) and the average processing time by 16.37 (ms),0.36 (ms), 58.37(ms) than ABC, ACO and PSO algorithms respectively.

Experiment 5:

In this experiment, each machine has different number of processor and speed. In DC1 3 physical hardware, each hosts have 16, 4 and 6 number of core processors and 4000, 6000 and 10000MIPS processor speeds respectively. In DC2 2 physical hardware, each hosts have 4 and 8 number of core processors and 4000 and 6000 MIPS processor speeds respectively. In DC3 4 physical hardware, each hosts have 8,2, 6 and 4 number of core processors and 4000, 6000,10000 and 8000MIPS processor speeds respectively. In DC4 2 physical hardware, each hosts have 2 and 4 number of core processors and 4000 and 6000 MIPS processor speeds. In DC5 3 physical hardware, each hosts have 6, 8 and 4 number of core processors and 4000, 6000 and 10000MIPS processor speeds respectively. The user base and application deployment configuration file is as experiment 4 table 5.12 and experiment 1 table 5.4.

Result

From this experiment, we obtain the following result in table 5.14:

Table 5.14: Experiment 5 result

Algorithms		Average (ms.)	Minimum(ms)	Maximum(ms)
ABC	Response time	139.69	40.46	7728.00
	Data Center processing time	32.99	0.23	7678.25
ACO	Response time	110.25	40.41	376.06
	Data Center processing time	6.63	0.15	41.63
PSO	Response time	206.37	43.24	19966.45
	Data Center processing time	98.34	0.89	19920.06
PSACO	Response time	109.78	40.25	445.71
	Data Center processing time	6.17	0.21	42.38

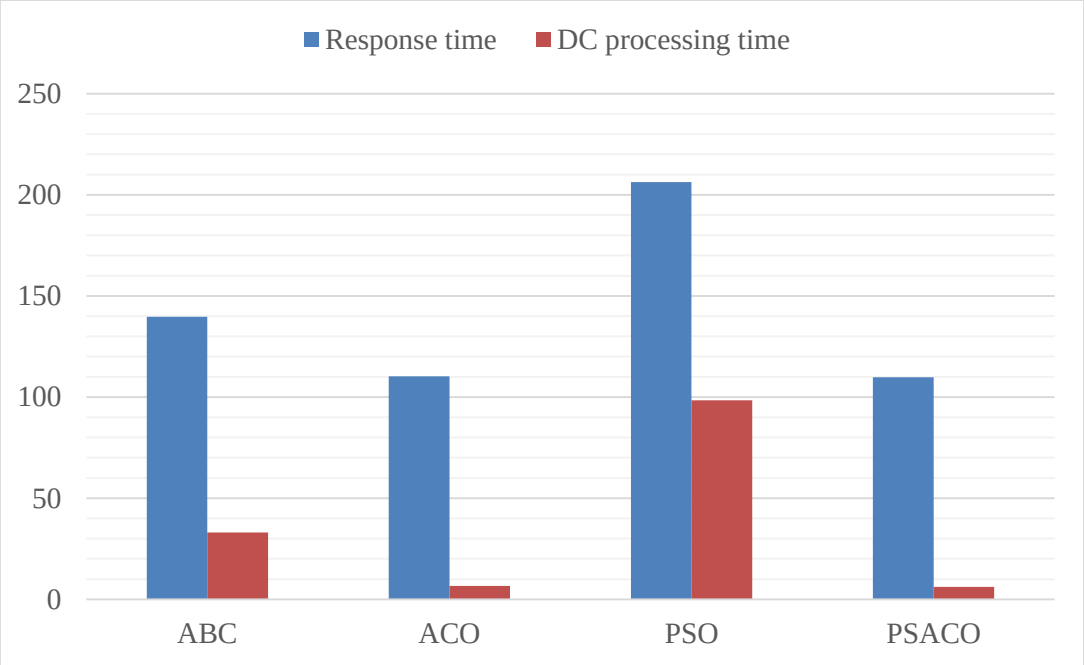


Figure 5.17 All algorithm result based on different number of processor and speed in heterogeneous environment

In the experiment 5.5, the above tables and figures shows that the propose hybrid algorithm has a better response time and processing time than the existing ABC, ACO and PSO algorithms based on different number of processor and speed in heterogeneous environment. The average response time of PSACO is 109.78 (ms) and the average processing time is 6.17 (ms). This result better than ABC, ACO and PSO algorithms, the average response time of ABC is 139.69 (ms) and the average processing time is 32.99 (ms). The average response time of ACO is 110.25 (ms), PSO is 206.37and the average processing time of ACO is 6.63 (ms), PSO is 98.34. The proposed algorithm minimizes the average response time by 29.91(ms) ,0.47 (ms), 96.59(ms) and the average processing time by 26.82 (ms),0.46 (ms), 88.17(ms) than ABC, ACO and PSO algorithms respectively.

Table 5.15: Data center request service time

	ABC	ACO	PSO	PSACO
D1	23.13	4.23	94.56	3.38
D2	17.74	5.72	20.06	5.27
D3	29.75	4.68	33.37	4.49
D4	26.43	21.13	14.27	22.04
D5	68.46	15.58	274.59	3.419

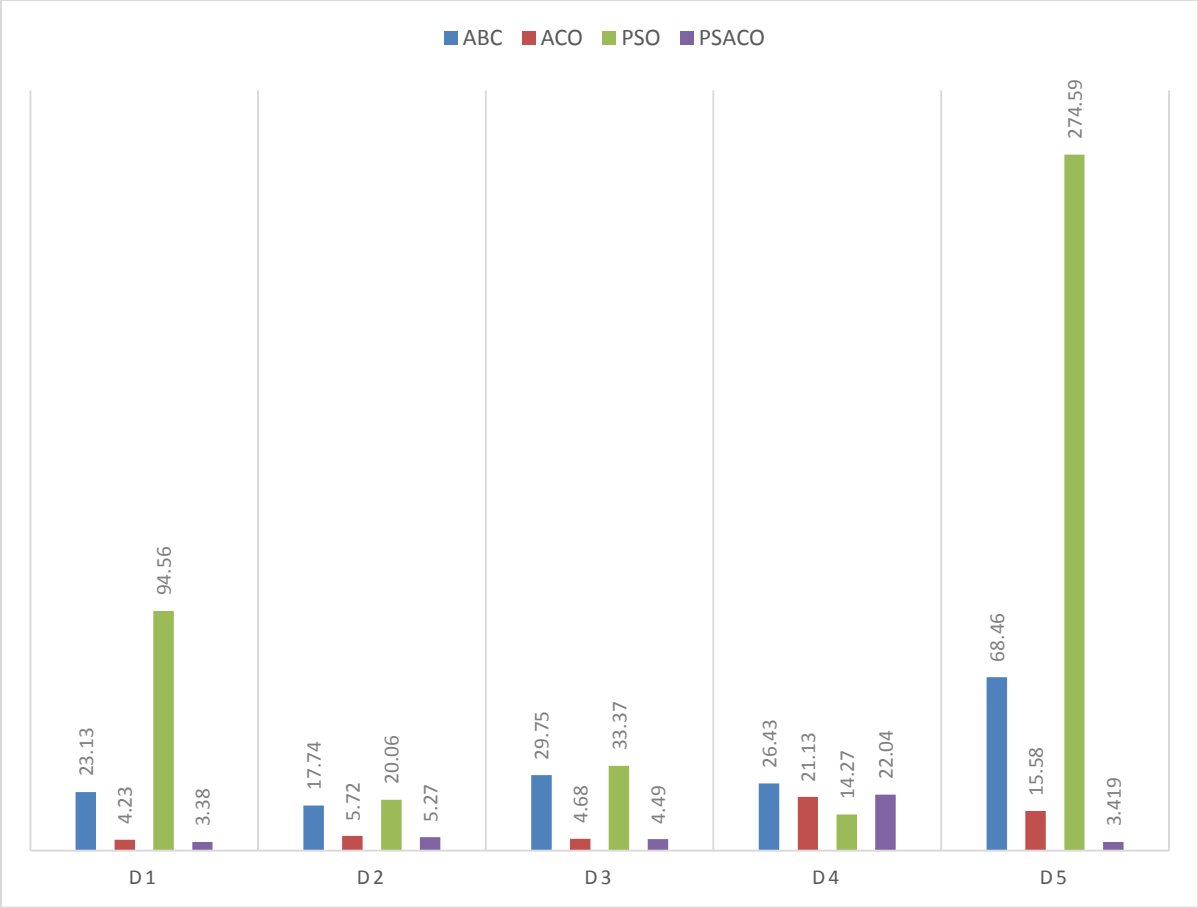


Figure 5.18 Data center request service time

Based on the above figure 5.8, the proposed algorithm has minimum requesting service time in datacenter 1,2,3 and 5 than other existing algorithm. PSO algorithm has minimum requesting service in datacenter 4 than other algorithms.

Experiment 6:

In this experiment**,** all configuration file is the same as experiment 5, but cost per VM per hour and data transfer cost of each DCs are not the same.

Table 5.16: Data center cost

Name	Cost per VM $/hour	Data Transfer Cost $/GB
DC1	0.9	0.1
DC2	0.8	0.2
DC3	0.7	0.3
DC4	0.6	0.4
DC5	0.5	0.5

Result

From this experiment, we obtain the following result in table 5.17:

Table 5.17: Experiment 6 response time and processing time result

Algorithms		Average (ms.)	Minimum(ms)	Maximum(ms)
ABC	Response time	139.62	40.46	7728.00
	Data Center processing time	32.70	0.23	7678.25
ACO	Response time	110.14	40.34	377.15
	Data Center processing time	6.52	0.15	41.63
PSO	Response time	195.67	43.24	15408.19
	Data Center processing time	87.57	0.89	15363.11
PSACO	Response time	109.64	40.34	374.62
	Data Center processing time	5.93	0.14	41.22

Table 5.18: Experiment 6 cost result

Algorithms	Data transfer cost$	Processing cost$
ABC	55.77	219.30
ACO	52.76	216.29
PSO	55.63	219.16
PSACO	52.69	216.22

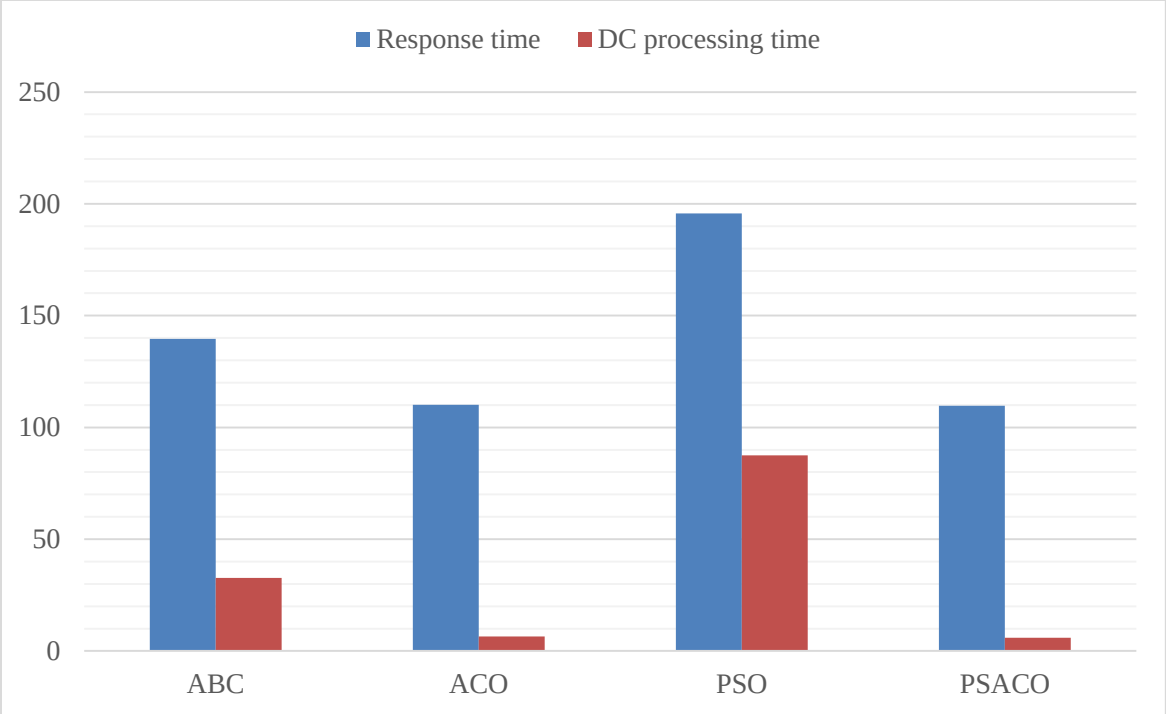


Figure 5.19 Response time and DC processing time result based on different DC cost

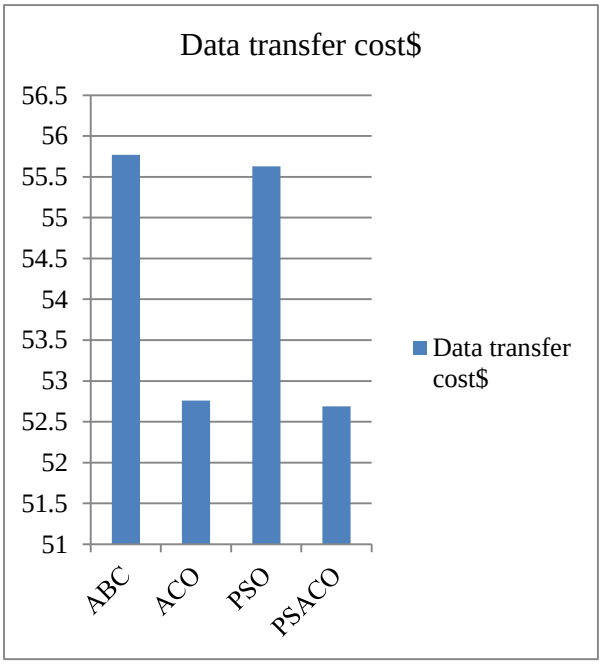


Figure 5.20 Transfer cost

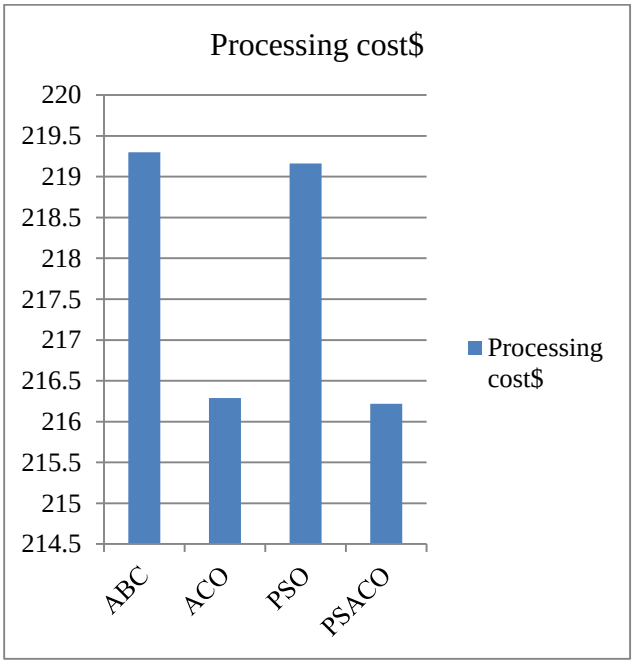


Figure 5.21 Processing cost

Based on experiment 6 result, the proposed algorithm has better response time, data center processing time, data transfer cost and processing cost than ABC, ACO and PSO.

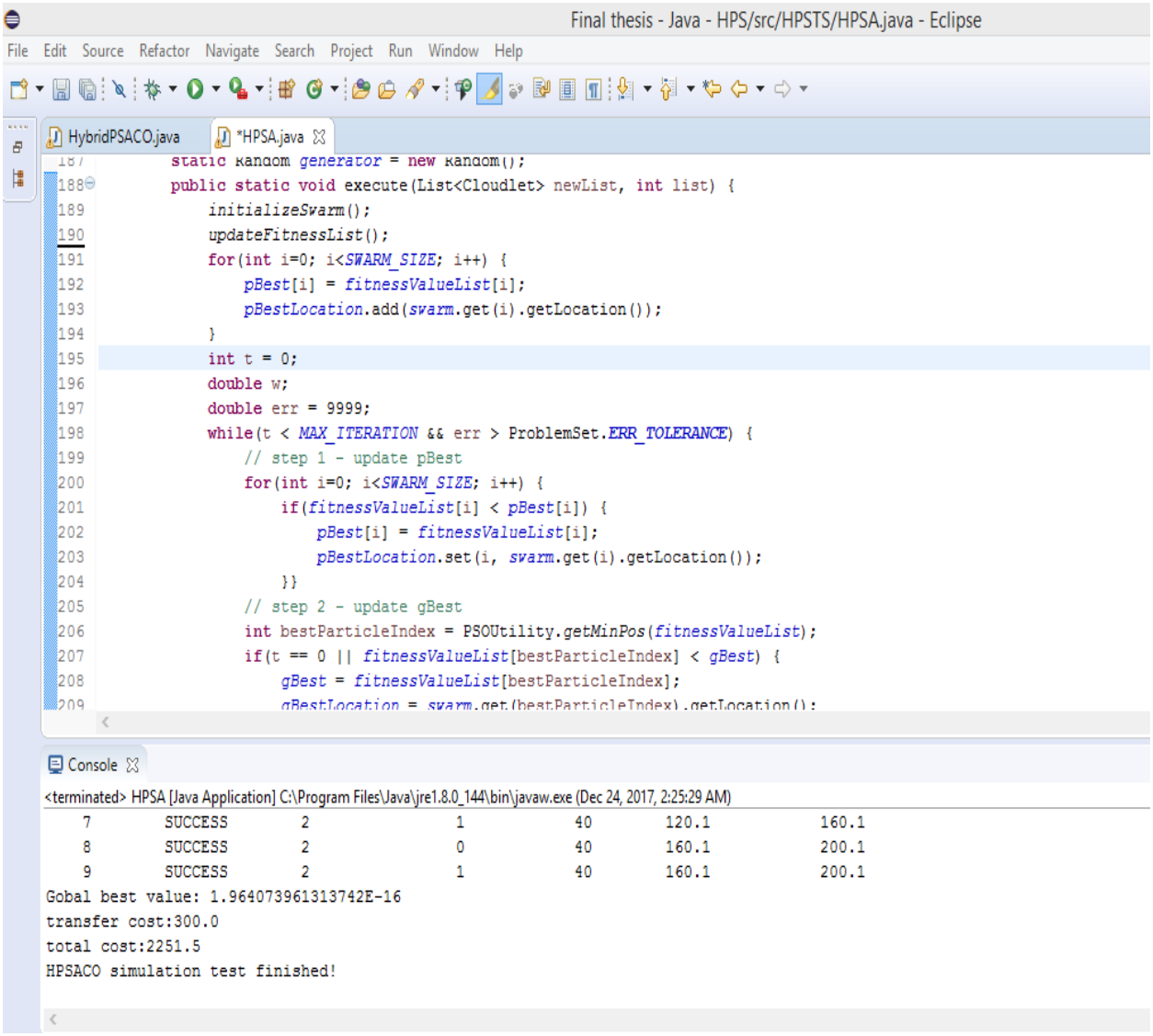


Figure 5.22: Fitness value simulation result

Figure 5.22 shows, how to get the global best fitness value from the total user request and assign this request to the selected VMs using cloudsim simulation tool.

During this evaluation, the best fittest value is calculating from the total swarm/cloudlets size until maximum iteration and assign the request to the available selected virtual machine

5.4. Summary

This chapter present implementation and experimental result of the proposed algorithm in order to address research question. The proposed algorithm evaluates in homogeneous and heterogeneous environment with the same number of cloud resource as well as different number of a cloud resource using cloud analyst simulator. The hybrid algorithm experimental result compare with three basic meta-heuristic algorithms. I.e. ABC, ACO and PSO algorithms.

In the first experiment, the algorithms evaluate in a single location with the same number of cloud resource based on response time and DC processing time parameters. Therefore, the proposed hybrid algorithm has a better response time and processing time than ABC, ACO and PSO. The proposed algorithm minimizes the average response time by 89.34(ms) and the average processing time by 67.12 (ms) than ABC algorithm. And also in the second experiment, the proposed algorithm shows performance improvement over existing algorithm when the application deployed in different location with different number of cloud resource. Such as, the proposed algorithm minimizes the average response time by 43.63(ms) and the average processing time by 38.06 (ms) than ABC algorithm. In experiment three and four, the number of tasks become larger than experiment 1&2 and the application is deployed in heterogeneous environment with different cloud resource. In both experimental result, the proposed hybrid algorithm has an optimum response and processing time than three existing algorithms.

In experiment five, the algorithms evaluate in different environment with different number of cloud resource and the capacity of processor is more powerful. And the number of processor and processor speed is different in each data centers. The proposed algorithm minimizes the average response time by 29.91(ms) ,0.47 (ms), 96.59(ms) and the average processing time by 26.82 (ms),0.46 (ms), 88.17(ms) than ABC, ACO and PSO algorithms respectively. Finally, in the experiment six, considering different cost per VM per hour and data transfer cost the proposed algorithm has better response time, data center processing time, data transfer cost and processing cost than ABC, ACO and PSO

In general, based on the above six experiment the proposed algorithms algorithm enhances a significant performance improvement on response time, processing time, data transfer cost and processing cost than other algorithms.

CHAPTER SIX

6. CONCLUSION AND FUTURE WORK

6.1. Conclusion

Cloud computing is an evolving concept which makes better use of multiple distributed resources that can be allocated to users as per requirements. This helps in cheaper and efficient utilization of available resources and easier handling of larger computational problems. Its advantages include transparency of resources, flexibility, location independence, reliability, affordability, and greater availability of services, etc. To provide these facilities, the tasks need to be scheduled properly on the resources so as to provide maximum performance in minimum time [12].

This resource allocation process is performed in two stages. In the first stage, the load balancer allocates resources to systems as requested by an application. The second stage takes place when incoming requests are assigned to an application to balance loads within the application as per QoS and minimum cost [36].

Task scheduling is an important issue in the cloud environment. There are several dynamic task scheduling approaches which are available in cloud environment to solve some part of task scheduling and load balancing problem. In the existing scheduling algorithm, still need an improvement to get more optimum response time, processing time and cost. Therefore, we proposed a hybrid task scheduling algorithm to improve the performance of cloud computing.

The proposed algorithm is considered an incorporation of the Particle Swarm Optimization and Ant Colony Optimization algorithms; we take the advantage of both algorithms to achieve the objective. The convergence speed of PSO algorithm is fast in the initial stage, but the local searching ability will decrease in the later due to the lack of population diversity. The convergence speed of ACO is slow due to the lack of pheromone in initial stage but later local searching ability is increase. The experiment is implemented using a cloudsim extension called cloud analyst simulation tool.

In the experiment, the proposed algorithm is tested based on response time, processing time transfer cost and processing cost evaluation metrics in six different experiment. The proposed algorithm is evaluated when the application is deployed in homogeneous environment and

heterogeneous environment. In the first experiment, the proposed algorithm is tested in a single location with the same number of cloud resource. On the others experiment, the proposed algorithm is tested in a heterogeneous environment with different cloud resource and different VM and transfer cost. Finally, compare the proposed hybrid algorithm result with the existing algorithm ABC, ACO and PSO. The result shows, the proposed algorithm has a better response time, processing time, data transfer cost and processing cost than three meta-heuristic algorithms.

6.2. Future Work

Task Scheduling is one of the major issue in cloud computing. We proposed a hybrid algorithm to improve the performance of cloud computing resources in a single location and heterogeneous location based on response time, processing time and resource utilization. But, still need an improvement to assign user requests to multiple cloud resources based on other parameters and factors such as, makespan time, storage cost, memory cost etc. In this study, we only consider scheduling issues in cloud computing, but there are other issues we are going to study in the future by using this hybrid meta-heuristic algorithm, such as travel sales man problem for vehicle routing system.

REFERENCE

[1] Bhathiya Wickremasinghe, Rodrigo N. Calheiros, and Rajkumar Buyya, (2010).” CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications”, 24th IEEE International Conference on Advanced Information Networking and Applications.

[2] Cse, M Tech., (2012). “Comparison of Load Balancing Algorithms in a Cloud Jaspreet Kaur.” International Journal of Engineering Research and Application,2(3): 1169–73.

[3] Sharma, Tejinder, Vijay Kumar Banga, (2013). “Efficient and Enhanced Algorithm in Cloud Computing.” International Journal of Soft Computing and Engineering 3(1): 385–90.

[4] Er. Kaur, N., Er. Nagpal, P., Singh, T., (2015, Sep). “A Review on Load Balancing Techniques in Cloud Computing “International Journal for Advance Research in Engineering and Technology, Volume 3, Issue IX.

[5] Ali Al-maamari, Fatma A. Omara, (2015).” Task Scheduling using Hybrid Algorithm in Cloud Computing Environments”, IOSR Journal of Computer Engineering(IOSR-JCE), Volume 17, Issue 3, PP 96-106.

[6] Kruekaew, B. and Kimpan, W., (2014)*.”* Virtual Machine Scheduling Management on Cloud Computing Using Artificial Bee Colony”, International Multi Conference of Engineers and Computer Scientists(IMECS), Volume 1, ISBN: 978-988-19252-5-1.

[7] Rastkhadiv, F., Zamanifar, K., (2016). “Task Scheduling Based on Load Balancing Using Artificial Bee Colony in Cloud Computing Environment”, International Journal of Advanced Biotechnology and Research (IJBR), Volume 7, Issue 5, pp. 1058-1069, ISSN 0976-2612, Online ISSN 2278–599X.

[8] Bidarkar, B., Attikeri, S., Ramya S. Pure, (2014).” Round Robin Approach for Better VM Load Balancing in Cloud computing”, International Journal of Engineering Innovation & Research (IJEIR), Volume 3 Issue 4, pp. 1-4, ISSN: 2277 – 5668.

[9] Verma, A., Kaushal, S., (2014). “Bi-Criteria Priority Based Particle Optimization Workflow Scheduling Algorithm for Cloud”, Proceedings of 2014 RAECS UIET Panjab University Chandigarh.

[10] Tawfeek, M., El-Sisi, A., Keshk, A., Torkey, T., (2015, March).” Cloud Task Scheduling Based on Ant Colony Optimization”, The International Arab Journal of Information Technology, Volume 12, Issue 2.

[11] Er. Mohinder Singh, Er. Ritu Marken, (2016, May). “A Survey on Task Scheduling Optimization in Cloud Computing”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 6, Issue, pp. 850-855.

[12] Bilgaiyan S. Sagnika S, Das M, (2014, March). “An Analysis of Task Scheduling in Cloud Computing using Evolutionary and Swarm-based Algorithms”, International Journal of Computer Applications (0975 – 8887), Volume 89, Issue 2.

[13] Shameer A.P, Dr. AC Subhajini, (2016, May). “Study on Different Scheduling Algorithm for Cloud Computing”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 6, Issue 5.

[14] Kumar, P., Anand, S. (2013)." An Approach to Optimize Workflow Scheduling for Cloud Computing Environment ", Journal of Theoretical and Applied Information Technology, Volume 57, Issue 3, pp. 1-7, ISSN: 1992-8645.

[15] Kaur, R., Luthra, P. (2014)." Load Balancing in Cloud Computing ", Proc. of Int. Conf. on Recent Trends in Information, Telecommunication and Computing, ITC.

[16] Lu Huang, Hai-shan Chen and Ting-ting Hu, (2013)."Survey on Resource Allocation Policy and Job Scheduling Algorithms of Cloud Computing”, Journal of Software, Volume 8, Issue 2, pp. 480-487.

[17] Kaur, R. Kinger, S., (2014).” Enhanced Genetic Algorithm based Task Scheduling in Cloud Computing” International Journal of Computer Applications, Volume 101, Issue 14.

[18] Himani, Singh Sidhu, H., (2014).” Comparative Analysis of Scheduling Algorithms of Cloudsim in Cloud Computing”, International Journal of Computer Applications, Volume 9, Issue 16.

[19] Manish Singh, R., Paul, S., Kumar, A., (2014).” Task Scheduling in Cloud Computing: Review” International Journal of Computer Science and Information Technologies(IJCSIT), Volume 5, Issue 6, pp.7940-7944.

[20] Bilgaiyan, S., Sagnika, S., Mishra, S., Das, M., (2015).” Study of Task Scheduling in Cloud Computing Environment Using Soft Computing Algorithms”, I.J. Modern Education and Computer Science, volume 3, pp32-38.

[21] Noble, H., & Smith, J. (2015). “Issues of validity and reliability in qualitative research”, Evidence-Based Nursing, Volume 18, Issue 2, pp.34-5, DOI: 10.1136/eb-2015-102054

[22] Nyaoro B. (2012). “A Framework to Guide Companies on Adopting Cloud Computing Technology”, Faculty of Information Technology, Strathmore University, Nairobi, Kenya.

[23] Assoc. Prof. Buyya, R., (2009).” CloudAnalyst: A CloudSim-based Tool for Modelling and Analysis of Large Scale Cloud Computing Environments”, Distributed Computing Project, CSSE Dept., University of Melbourne.

[24] B. Booba, Dr. T.V. GOPAL, (2013).” Comparison of Ant colony optimization and Particle Swarm optimization in grid environment”, International Journal of advance research in computer science and applications, Volume 1, Issue 5.

[25] James Kennedy and Russell Eberhart, (1995). “Particle Swarm Optimization”, IEEE.

[26] Patel, D., Mr. S Rajawat, A., (2015).” Efficient Throttled Load Balancing Algorithm in Cloud Environment”, International Journal of Modern Trends in Engineering and Research, Volume 02, Issue 03.

[27] Panwar, R., Mallick, B., (2015).” A Comparative Study of Load Balancing Algorithms in Cloud Computing”, International Journal of Computer Applications, Volume 117, Issue 24.

[28] Rani, P., Chauhan, R., Chauhan, R., (2015).” An Enhancement in Service Broker Policy for Cloud-Analyst”, International Journal of Computer Applications, Volume 115, Issue 12, pp. 0975 – 8887.

[29] Mishra, R.K. and Bhukya, S.N., (2014). “Service Broker Algorithm for Cloud-Analyst”, International Journal of Computer Science and Information Technologies, Volume 5, Issue 3, pp.3957-3962.

[30] Dr. Agarwal, A., Jain, S. (2014).” Efficient Optimal Algorithm of Task Scheduling in Cloud Computing environment”, International Journal of Computer Trends and Technology (IJCTT), Volume 9, Issue 7, pp. 344–349, ISSN: 2231-2803.

[31] Safwat A. Hamad, Fatma A. Omara, (2016).” Genetic-Based Task Scheduling Algorithm in Cloud Computing Environment”, International Journal of Advanced Computer Science and Applications, Volume 7, Issue 4, pp. 550-556.

[32] Pakize, R., S., khademi, M., S., Gandomi, A., (2014, May).” Comparison of CloudSim, CloudAnalyst and CloudReports Simulator in Cloud Computing”, International journal of Computer Science & Network Solutions, Volume 2, Issue 5, ISSN 2345-3397.

[33] R. Buyya, and M. Murshed, (2002). “GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing,” Concurrency and Computation: Practice and Experience, vol. 14, no. 13-15, pp. 1175-1220.

[34] Manoel C. Silva Filho, Raysa L. Oliveira, Claudio C. Monteiro, Ped, (2017, Oct)” CloudSim Plus Documentation”, Release 1.0.0.

[35] Bala, K., (2017, Jan).” A study on Particle Swarm Optimization Scheduling for cloud computing environment”, International Journal of Engineering Research and General Science Volume 5, Issue 1, ISSN 2091-2730.

[36] Patel, R., and Patel, S., (2013, Feb). "Survey on Resource Allocation Strategies in Cloud Computing”, International Journal of Engineering Research & Technology (IJERT) Volume 2 Issue 2, pp.1-5.

[37] Guo, L., Zhao, S., Shen, S., and Jiang, C., (2012, March). "Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm", Journal of Networks, Volume 7, Issue 3, pp.547-553.

[38] Isam Azawi Mohialdeen, (2013). “A Comparative Study of Scheduling Algorithms in Cloud Computing Environment” Science Publications.

[39] Ijaz S., Munir E., Anwar W., and Nasir W., (2013). “Efficient Scheduling Strategy for Task Graphs in Heterogeneous Computing Environment,” the International Arab Journal of Information Technology, volume 10, Issue 5, pp. 486-492.

[40] Alexander, A., A., Joseph, L., D., (2016). “An Efficient Resource Management for Prioritized Users in Cloud Environment Using Cuckoo Search Algorithm”, Global Colloquium in Recent Advancement and Effectual Researches in Engineering, Science and Technology.

[41] Shailendra S. Aote, (2013, Sep). “A Brief Review on Particle Swarm Optimization: Limitations & Future Directions” International Journal of Computer Science Engineering (IJCSE), Volume 2, Issue 5, ISSN: 2319-7323.

[42] Hevner, A., Chatterjee, S., (2010), “Design Research in Information Systems”, Theory and Practice, Volume 28, pp. 320.

Appendixes

Appendix 1: Sample code of the Hybrid PSACO algorithm

*/** The code that to find global best value. */*

```
public class PSA {
        private int SWARM_SIZE=dcbs.getVmStatesList().size();
        private Vector<Particle> swarm = new Vector<Particle>();
        private double[] pBest = new double[SWARM_SIZE];
        private Vector<Location> pBestLocation = new Vector<Location>();
        private double gBest;
        private Location gBestLocation;
        private double[] fitnessValueList = new double[SWARM_SIZE];
        Random generator = new Random();
        public int execute() {
                initializeSwarm();
                updateFitnessList();
                for(int i=0; i<SWARM_SIZE; i++) {
                        pBest[i] = fitnessValueList[i];
                        pBestLocation.add(swarm.get(i).getLocation());
                }
                int t = 0;
                double w;
                double err = 9999;
                while(t < MAX_ITERATION && err > ProblemSet.ERR_TOLERANCE) {

// step 1 update pBest
                        for(int i=0; i<SWARM_SIZE; i++) {
                                if(fitnessValueList[i] < pBest[i]) {
                                        pBest[i] = fitnessValueList[i];
                                        pBestLocation.set(i, swarm.get(i).getLocation());
                                }}
// step 2 update gBest
                        int bestParticleIndex = PSOUtility.getMinPos(fitnessValueList);
                        if(t == 0 || fitnessValueList[bestParticleIndex] < gBest) {
                                gBest = fitnessValueList[bestParticleIndex];
                                gBestLocation = swarm.get(bestParticleIndex).getLocation();
                        }
                        w = W_UPPERBOUND - (((double) t) / MAX_ITERATION) *
(W_UPPERBOUND - W_LOWERBOUND);
                        for(int i=0; i<SWARM_SIZE; i++) {
                                double r1 = generator.nextDouble();
                                double r2 = generator.nextDouble();
                                Particle p = swarm.get(i);
// step 3 update velocity
                                double[] newVel = new double[PROBLEM_DIMENSION];
```

```
                        newVel[0] = (w * p.getVelocity().getPos()[0]) +
                                        (r1 * C1) *
(pBestLocation.get(i).getLoc()[0] - p.getLocation().getLoc()[0]) +
                                        (r2 * C2) * (gBestLocation.getLoc()[0] -
p.getLocation().getLoc()[0]);
                        newVel[1] = (w * p.getVelocity().getPos()[1]) +
                                        (r1 * C1) *
(pBestLocation.get(i).getLoc()[1] - p.getLocation().getLoc()[1]) +
                                        (r2 * C2) * (gBestLocation.getLoc()[1] -
p.getLocation().getLoc()[1]);
                        Velocity vel = new Velocity(newVel);
                        p.setVelocity(vel);

// step 4 update location
                        double[] newLoc = new double[PROBLEM_DIMENSION];
                        newLoc[0] = p.getLocation().getLoc()[0] + newVel[0];
                        newLoc[1] = p.getLocation().getLoc()[1] + newVel[1];
                        Location loc = new Location(newLoc);
                        p.setLocation(loc);
                    }
                    err = ProblemSet.evaluate(gBestLocation) - 0;
                    t++;
                    updateFitnessList();
                }
                int temp =(int) ProblemSet.evaluate(gBestLocation);
                return temp;
        }
        public void initializeSwarm() {
                Particle p;
                for(int i=0; i<SWARM_SIZE; i++) {
                        p = new Particle();
                        double[] loc = new double[PROBLEM_DIMENSION];
loc[0] = ProblemSet.LOC_X_LOW + generator.nextDouble() * (ProblemSet.LOC_X_HIGH -
ProblemSet.LOC_X_LOW);
loc[1] = ProblemSet.LOC_Y_LOW + generator.nextDouble() * (ProblemSet.LOC_Y_HIGH -
ProblemSet.LOC_Y_LOW);
                        Location location = new Location(loc);
                        double[] vel = new double[PROBLEM_DIMENSION];
vel[0] = ProblemSet.VEL_LOW + generator.nextDouble() * (ProblemSet.VEL_HIGH -
ProblemSet.VEL_LOW);
vel[1] = ProblemSet.VEL_LOW + generator.nextDouble() * (ProblemSet.VEL_HIGH -
ProblemSet.VEL_LOW);
                        Velocity velocity = new Velocity(vel);
                        p.setLocation(location);
                        p.setVelocity(velocity);
                        swarm.add(p);
```

```
        }}
    public void updateFitnessList() {
        for(int i=0; i<SWARM_SIZE; i++) {
            fitnessValueList[i] = swarm.get(i).getFitnessValue();
        }}}
```

*/** The code that to select a suitable VM from all available virtual machine to allocate the incoming user request. */*

```
public class Ant {
    private double[][] pheromones;
    private boolean[] isVisited;
    public List<Integer> path;
    private int VmId;
    public Ant(double[][] ph) {
        pheromones = ph;
        isVisited = new boolean[pheromones.length];
        VmId = isVisited.length - 1;
        path = new ArrayList<Integer>();
    }
    public int SendAnt() {
        return ProcessAnt(true);
    }
    public int FetchFinalVm() {
        return ((VirtualMachine) dcbLocal.vmlist.get(ProcessAnt(false ))).getVmId();
    }
    public int ProcessAnt(boolean updatePheromones) {
        int CurrentVmId = VmId;
        int nextVmId = getNextVmNode(CurrentVmId);
        if (updatePheromones) {
            UpdatePheromone(CurrentVmId, nextVmId);
        }
        while (nextVmId != CurrentVmId) {
            path.add(nextVmId);
            CurrentVmId = nextVmId;
            nextVmId = getNextVmNode(CurrentVmId);
            if (updatePheromones) {
                UpdatePheromone(CurrentVmId, nextVmId);
            }}
        if (updatePheromones) {
            UpdateGlobalPheromones();
        }
        return CurrentVmId;
    }
    public int getNextVmNode(int vmId) {
        double[] probability = computeProbability(vmId);
```

```
        Random rand = new Random();
        double randomization = rand.nextDouble();
        for (int i = 0; i < probability.length; i++) {
                randomization = randomization - probability[i];
                if (randomization <= 0) {
                        return i;
                }}
        for (int i = 0; i < probability.length; i++) {
                System.out.println("Debug " + probability[i]);
        }
        return -1;
}
public double[] computeProbability(int vmId) {
        double[] probability = new double[pheromones.length - 1];
        double sum = 0.0;
        for (int i = 0; i < probability.length; i++) {
                if (isVisited[i]) {
                        probability[i] = 0;
                        continue;}
                probability[i] = scoreFunction(vmId, i);
                sum += probability[i];
        }
        for (int i = 0; i < probability.length; i++) {
                probability[i] = probability[i] / sum;
        }
        return probability;
}
public void UpdatePheromone(int prevId, int newId) {
        pheromones[prevId][newId] += ONE_UNIT_PHEROMONE;
}

public double scoreFunction(int prevVmId, int newVmId) {
         double maxBw = ((VirtualMachine) dcbLocal.vmlist.get(newVmId))
                        .getCharacteristics().getBw();
        double currentBw = ((VirtualMachine) dcbLocal.vmlist.get(newVmId))
                        .getBw();
        return Math.pow(pheromones[prevVmId][newVmId], alpha) + 1.0
                        + (maxBw - currentBw / maxBw);
}
public void UpdateGlobalPheromones()
{}}}
```

Appendix 2: sample Snapshots

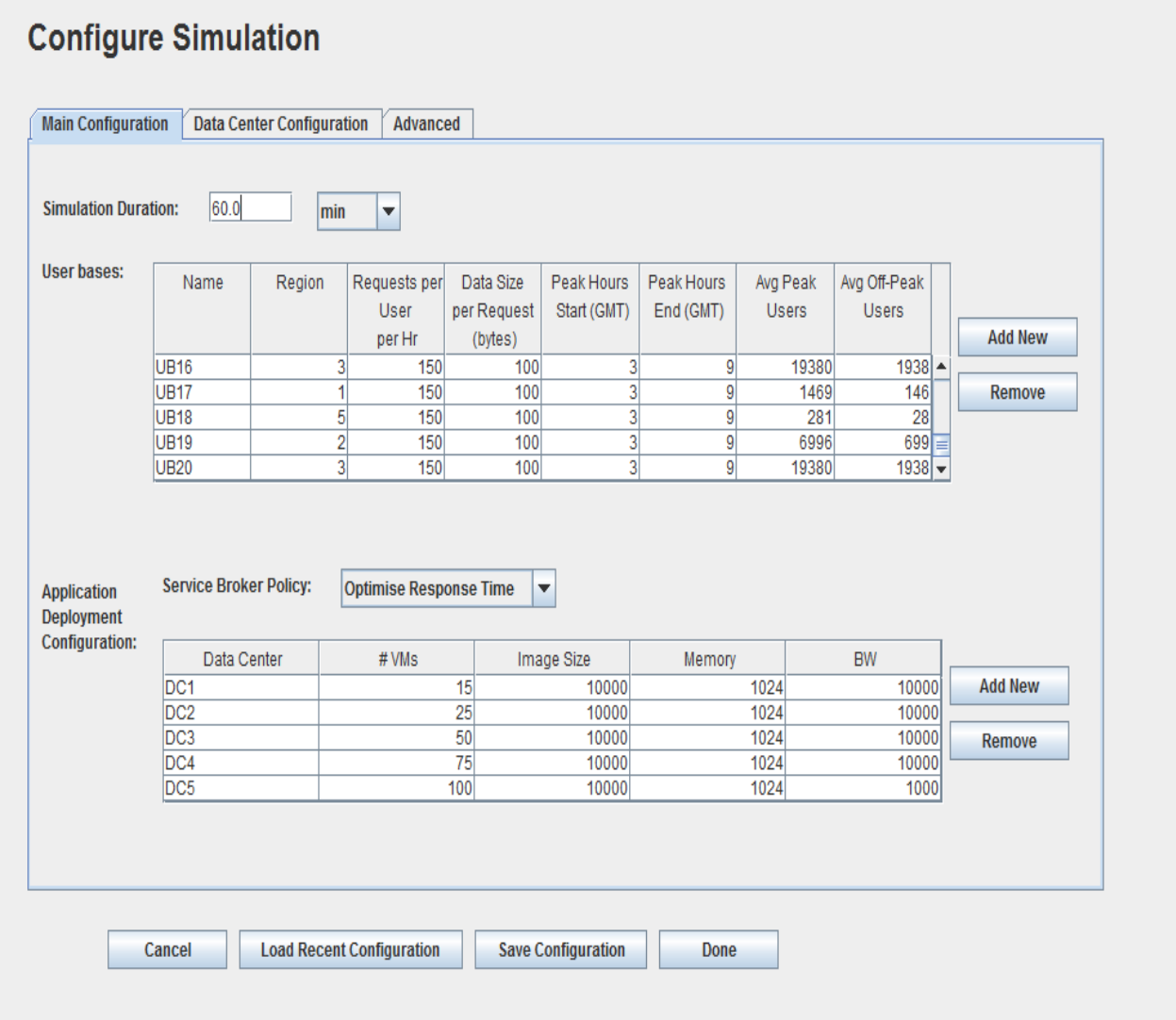


Fig1: Configure Simulation using 20 UB.

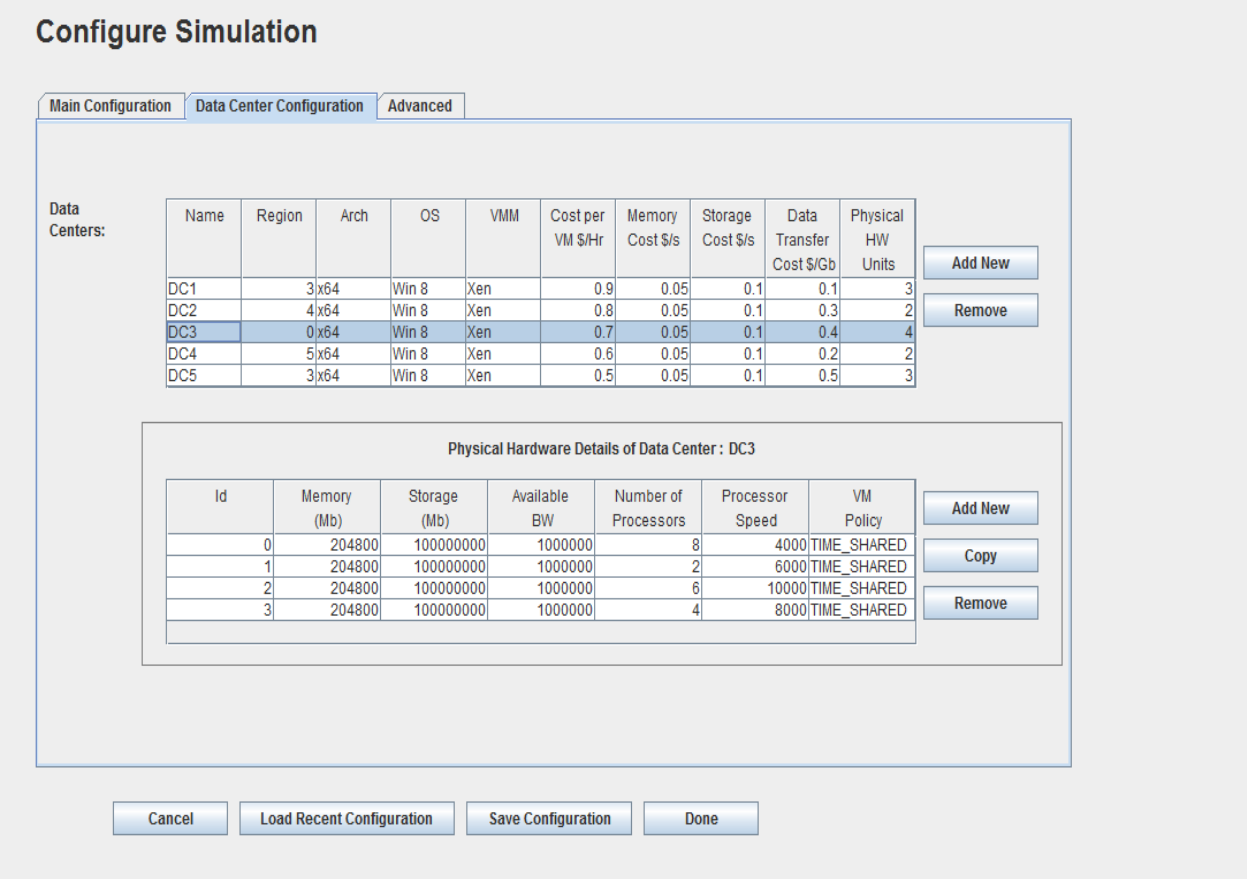


Fig 2: Data center configuration and Physical Hardware details

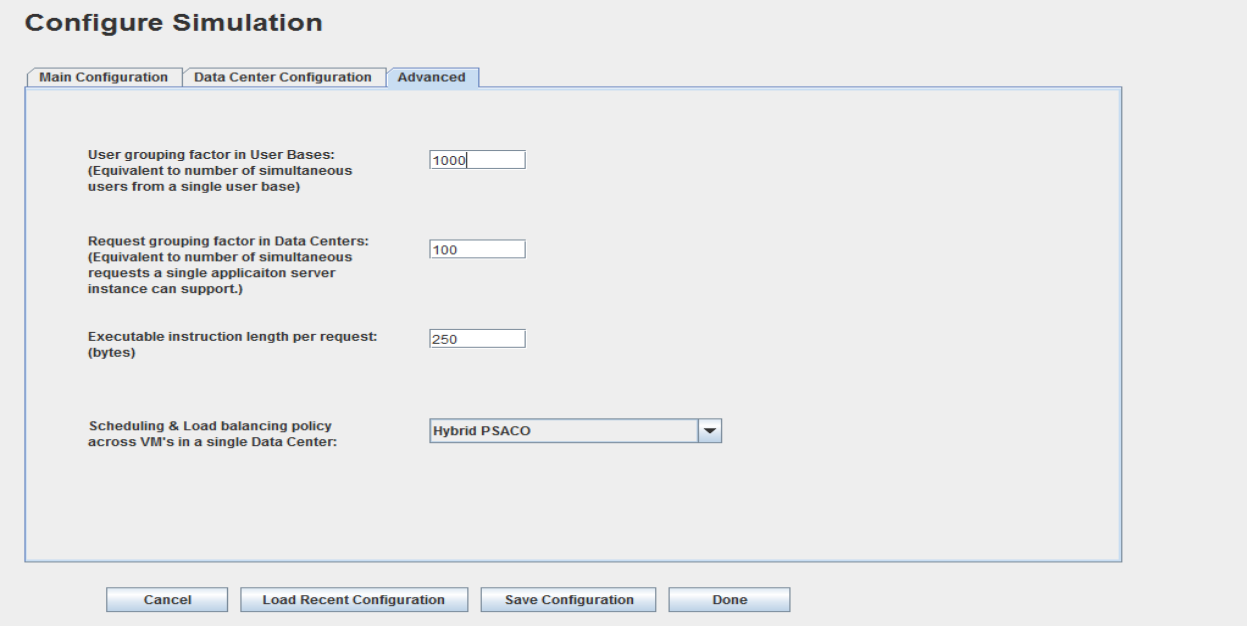


Fig 3: Select hybrid PSACO scheduling algorithm

Overall Response Time Summary

Export Results

	Average (ms)	Minimum (ms)	Maximum (ms)
Overall Response Time:	109.64	40.34	374.62
Data Center Processing Time:	5.93	0.14	41.22

Response Time By Region

Userbase	Avg (ms)	Min (ms)	Max (ms)
UB10	54.762	41.959	63.462
UB11	205.082	168.651	237.519
UB12	56.053	43.238	75.535
UB13	304.861	250.45	374.616
UB14	54.53	42.361	67.369
UB15	71.23	50.502	91.097
UB16	56.049	42.169	75.389
UB17	205.203	167.354	248.622
UB18	71.414	52.482	94.722
UB19	303.167	250.383	371.938
UB1	53.807	40.342	68.723
UB20	57.105	44.196	79.361
UB2	203.238	163.443	235.154
UB3	302.936	241.407	353.525
UB4	56.238	43.19	76.798
UB5	55.118	43.431	65.32
UB6	72.091	51.46	100.253
UB7	305.077	253.704	373.356
UB8	71.348	50.898	96.94
UB9	54.108	45.695	63.629

Fig 4: Overall Response Time and Data center processing time summary for proposed algorithm

Data Center Request Servicing Times

Data Center	Avg (ms)	Min (ms)	Max (ms)
DC1	3.434	0.143	13.112
DC2	5.31	0.3	12.004
DC3	4.541	0.141	15.016
DC4	22.039	8.709	41.22
DC5	14.798	0.506	26.263

Data Center Hourly Average Processing Times

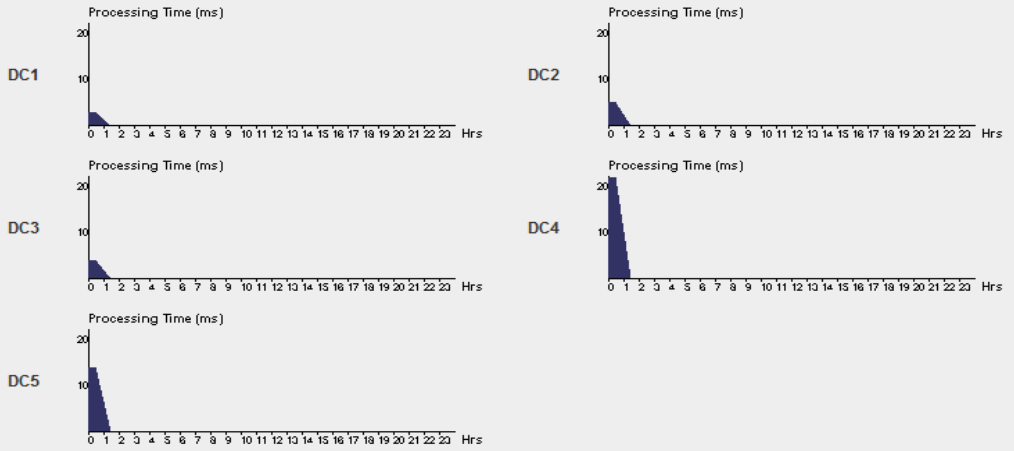
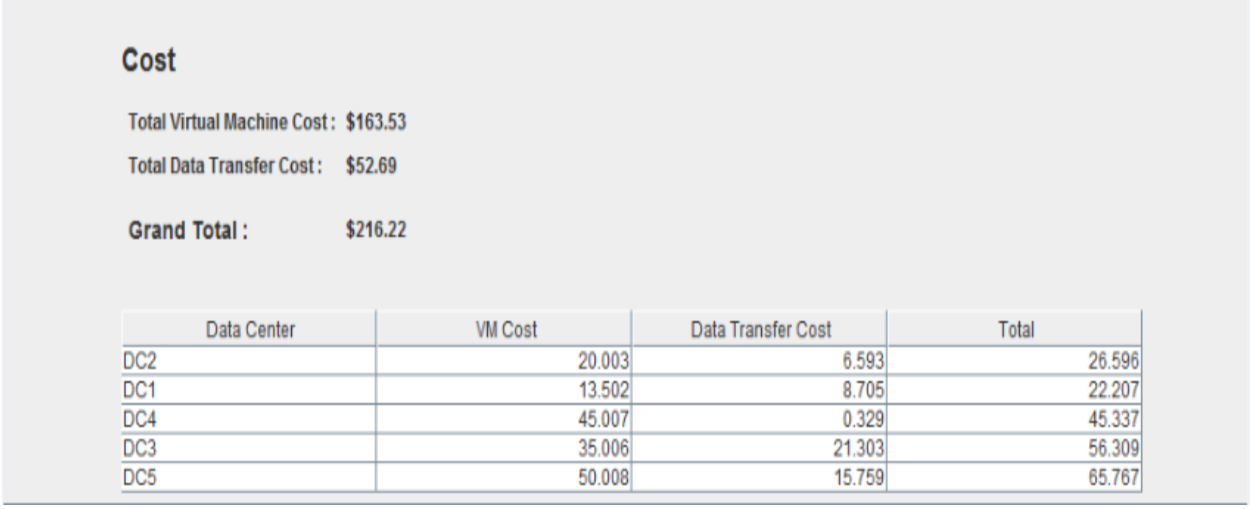


Fig 5: DC requesting service time summary for proposed algorithms



Cost

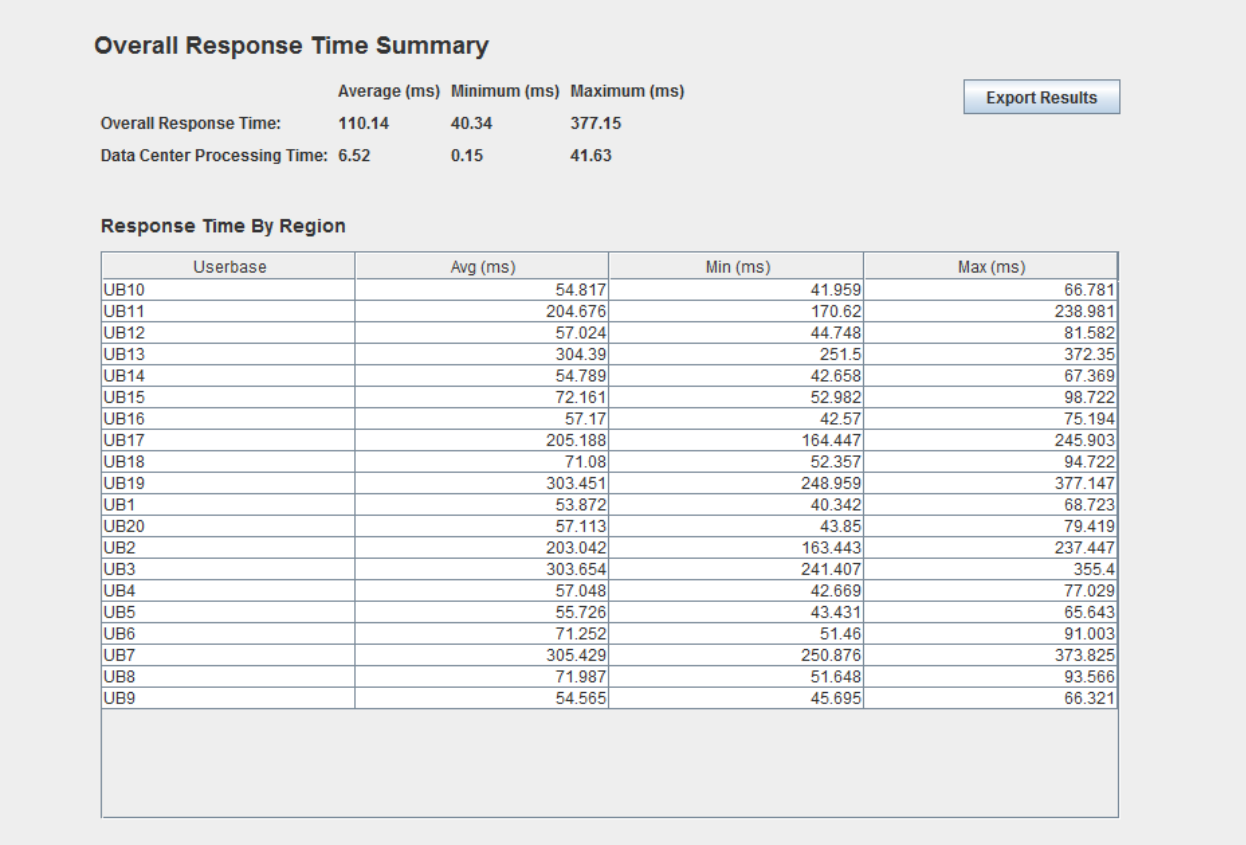
Total Virtual Machine Cost : $163.53

Total Data Transfer Cost : $52.69

Grand Total : $216.22

Data Center	VM Cost	Data Transfer Cost	Total
DC2	20.003	6.593	26.596
DC1	13.502	8.705	22.207
DC4	45.007	0.329	45.337
DC3	35.006	21.303	56.309
DC5	50.008	15.759	65.767

Fig 6: Cost Summary for proposed algorithm



Overall Response Time Summary

	Average (ms)	Minimum (ms)	Maximum (ms)
Overall Response Time:	110.14	40.34	377.15
Data Center Processing Time:	6.52	0.15	41.63

Export Results

Response Time By Region

Userbase	Avg (ms)	Min (ms)	Max (ms)
UB10	54.817	41.959	66.781
UB11	204.676	170.62	238.981
UB12	57.024	44.748	81.582
UB13	304.39	251.5	372.35
UB14	54.789	42.658	67.369
UB15	72.161	52.982	98.722
UB16	57.17	42.57	75.194
UB17	205.188	164.447	245.903
UB18	71.08	52.357	94.722
UB19	303.451	248.959	377.147
UB1	53.872	40.342	68.723
UB20	57.113	43.85	79.419
UB2	203.042	163.443	237.447
UB3	303.654	241.407	355.4
UB4	57.048	42.669	77.029
UB5	55.726	43.431	65.643
UB6	71.252	51.46	91.003
UB7	305.429	250.876	373.825
UB8	71.987	51.648	93.566
UB9	54.565	45.695	66.321

Fig 7: Overall Response Time and Data center processing time summary for ACO algorithm

Data Center Request Servicing Times

Data Center	Avg (ms)	Min (ms)	Max (ms)
DC1	4.216	0.165	20.567
DC2	5.629	0.25	17.408
DC3	4.677	0.146	21.268
DC4	22.092	8.709	41.627
DC5	15.823	0.956	33.576

Data Center Hourly Average Processing Times

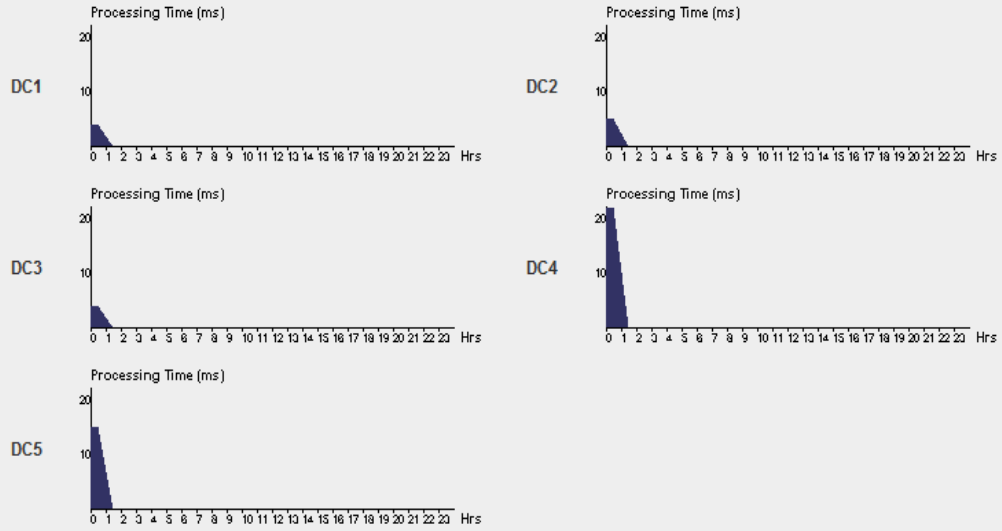


Fig 8: DC requesting service time summary for ACO algorithms

Cost

Total Virtual Machine Cost : $163.53

Total Data Transfer Cost : $52.76

Grand Total : $216.29

Data Center	VM Cost	Data Transfer Cost	Total
DC2	20.003	6.476	26.479
DC1	13.502	8.696	22.198
DC4	45.007	0.329	45.337
DC3	35.006	21.457	56.462
DC5	50.008	15.807	65.815

Fig 9: Cost Summary for ACO algorithm

Overall Response Time Summary

	Average (ms)	Minimum (ms)	Maximum (ms)
Overall Response Time:	195.67	43.24	15408.19
Data Center Processing Time:	87.57	0.89	15363.11

Export Results

Response Time By Region

Userbase	Avg (ms)	Min (ms)	Max (ms)
UB10	66.133	45.168	135.387
UB11	213.776	175.276	425.735
UB12	215.454	44.853	15,408.188
UB13	348.396	271.157	711.673
UB14	65.843	43.236	96.644
UB15	114.731	50.502	7,728.003
UB16	168.26	43.32	972.147
UB17	214.024	177.781	400.28
UB18	63.076	52.482	194.63
UB19	340.429	266.883	451.272
UB1	103.901	43.459	390.998
UB20	174.477	45.46	1,970.434
UB2	216.789	169.526	435.734
UB3	348.865	265.64	688.512
UB4	167.129	46.691	1,012.188
UB5	67.994	44.009	108.235
UB6	63.172	51.794	207.755
UB7	342.917	268.743	516.477
UB8	65.226	50.898	223.63
UB9	101.294	48.818	513.17

Fig 10: Overall Response Time and Data center processing time summary for PSO algorithm

Data Center Request Servicing Times

Data Center	Avg (ms)	Min (ms)	Max (ms)
DC1	102.355	1.061	15,363.113
DC2	19.941	0.886	83.286
DC3	33.744	1.261	349.958
DC4	27.807	6.397	7,678.251
DC5	188.634	2.85	959.196

Data Center Hourly Average Processing Times

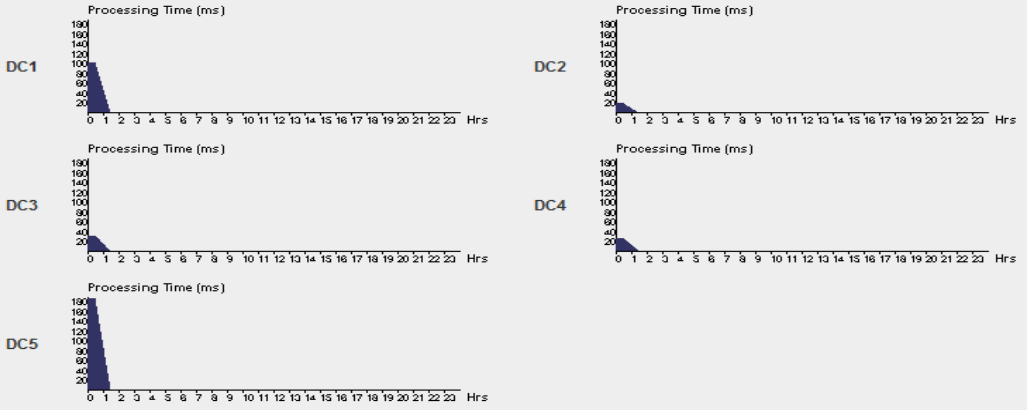


Fig 11: DC requesting service time summary for PSO algorithms

Cost

Total Virtual Machine Cost : $163.53

Total Data Transfer Cost : $55.63

Grand Total : $219.16

Data Center	VM Cost	Data Transfer Cost	Total
DC2	20.003	8.198	28.201
DC1	13.502	7.63	21.132
DC4	45.007	0.587	45.595
DC3	35.006	20.907	55.912
DC5	50.008	18.307	68.315

Fig 12: Cost Summary for ACO algorithm

Overall Response Time Summary

	Average (ms)	Minimum (ms)	Maximum (ms)
Overall Response Time:	139.62	40.46	7728.00
Data Center Processing Time:	32.70	0.23	7678.25

Export Results

Response Time By Region

Userbase	Avg (ms)	Min (ms)	Max (ms)
UB10	65.455	45.168	103.887
UB11	212.537	175.276	425.735
UB12	87.688	45.279	285.765
UB13	342.241	271.157	527.476
UB14	65.056	43.236	89.026
UB15	113.695	50.502	7,728.003
UB16	85.961	43.32	217.287
UB17	211.064	175.404	251.778
UB18	61.327	52.482	96.012
UB19	336.626	258.639	446.794
UB1	90.787	40.459	362.113
UB20	87.368	45.033	283.932
UB2	211.622	169.526	425.904
UB3	340.394	256.89	585.63
UB4	86.097	43.159	198.487
UB5	67.582	44.009	99.649
UB6	61.278	52.21	94.259
UB7	340.828	254.371	543.606
UB8	64.551	50.898	236.506
UB9	98.442	47.577	513.17

Fig 13: Overall Response Time and Data center processing time summary for ABC algorithm

Data Center Request Servicing Times

Data Center	Avg (ms)	Min (ms)	Max (ms)
DC1	22.822	0.232	50.817
DC2	17.688	0.491	54.238
DC3	30.309	0.418	98.63
DC4	26.557	4.044	7,678.251
DC5	65.223	0.493	232.965

Data Center Hourly Average Processing Times

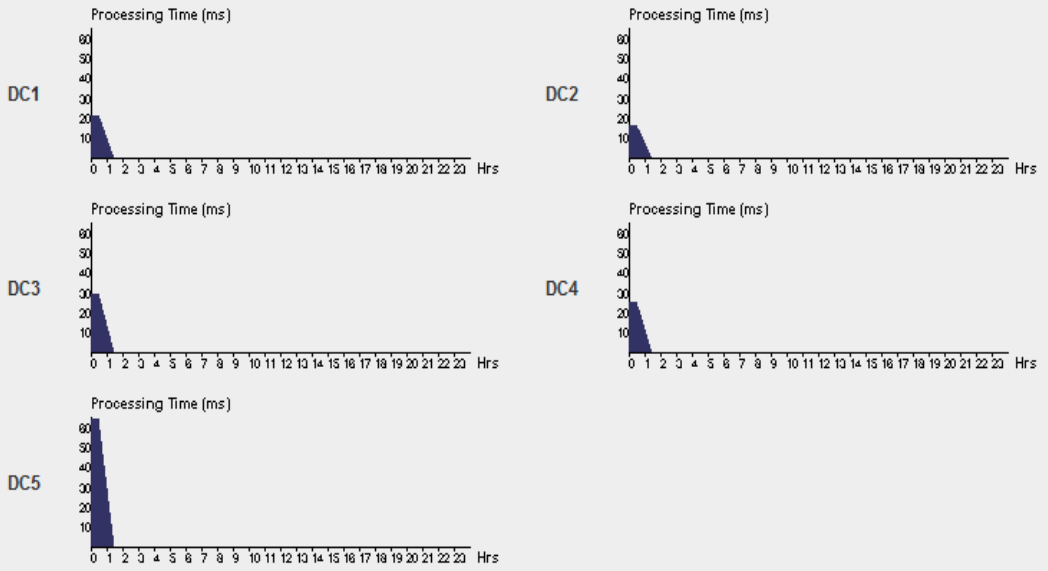


Fig 14: DC requesting service time summary for ABC algorithms

Cost

Total Virtual Machine Cost : $163.53

Total Data Transfer Cost : $55.77

Grand Total : $219.30

Data Center	VM Cost	Data Transfer Cost	Total
DC2	20.003	7.031	27.034
DC1	13.502	7.823	21.326
DC4	45.007	0.479	45.486
DC3	35.006	20.996	56.002
DC5	50.008	19.446	69.454

Fig 15: Cost Summary for ABC algorithm

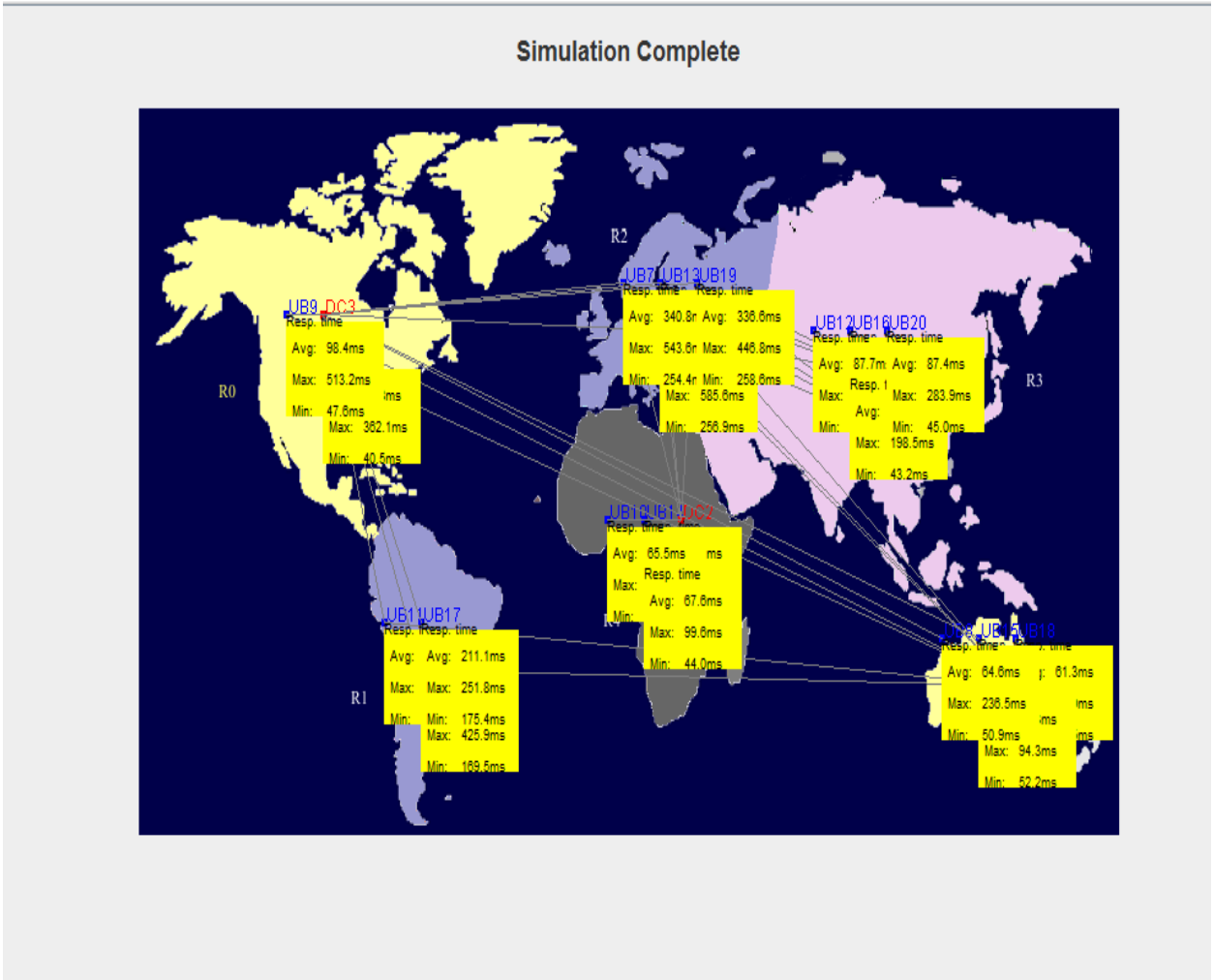


Fig 16: simulation result screen in different location