Tribhuvan University

Institute of Science and Technology

Task Scheduling in Cloud Computing Environment using Evolutionary and Swarm Based Algorithm

Dissertation

Submitted to

Central Department of Computer Science and Information Technology

Tribhuvan University, Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements

for the Master's Degree in Computer Science and Information Technology

By

Nawa Raj Rimal

November, 2017

Tribhuvan University

Institute of Science and Technology

Task Scheduling in Cloud Computing Environment using Evolutionary and Swarm Based Algorithm

Dissertation

Submitted to

Central Department of Computer Science and Information Technology

Tribhuvan University, Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements

for the Master's Degree in Computer Science and Information Technology

By

Nawa Raj Rimal

November, 2017

Supervisor

Prof. Dr. Subarna Shakya

Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science and Information Technology

Student's Declaration

I hereby declare that I am the only author of this work and that no sources other than the listed here have been used in this work.

… … … … … … …
Nawa Raj Rimal
November, 2017

Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science and Information Technology

Supervisor's Recommendation

We hereby recommend that this dissertation prepared under our supervision by **Mr. Nawa Raj Rimal** entitled **"Task Scheduling in Cloud Computing Environment using Evolutionary and Swarm Based Algorithm**" be accepted as partial fulfillment of the requirements for the degree of M. Sc. in Computer Science and Information Technology. In our best knowledge, this is an original work in computer science.

……..

Prof. Dr. Subarna Shakya
Department of Electronics & Computer Engineering
Institute of Engineering (IOE)
Tribhuvan University
Pulchwok, Kathmandu, Nepal
(Supervisor)

Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science and Information Technology

LETTER OF APPROVAL

We certify that we have read this dissertation and in our opinion, it is satisfactory in the scope and quality as a dissertation in the partial fulfillment for the requirement of Masters Degree in Computer Science and Information Technology.

Evaluation Committee

……..
Prof. Dr. SubarnaShakya
Department of Electronics & Computer Engineering
Institute of Engineering (IOE)
Tribhuvan University
Pulchwok, Kathmandu, Nepal
(Supervisor)

……..
Asst. Prof.NawarajPaudel
Central Department of Computer Science and
Information Technology (CDCSIT)
Tribhuvan University
Kritipur
Head of Department (HOD)

(External Examiner)

(Internal Examiner)

Acknowledgements

I would like to express my gratitude to all the people who supported and accompanied me during the preparation of this dissertation **"Task Scheduling in Cloud Computing Environment using Evolutionary and Swarm Based Algorithm"**. This research work has been performed under Central Department of Computer Science and Information Technology (Tribhuwan University), Kirtipur. I am very grateful to my department for giving me an enthusiastic support.

First, I would like to express my gratitude to my supervisor Professor Dr. Subarna Shakya, Professor of the Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk. This research would not have been possible without his advices and patience.

Most importantly I would like to thank to respected Head of Department of Central Department of Computer Science and Information Technology, Assoc. Prof. Nawaraj Poudel, respected teachers of CDCSIT, TU, for providing me such a broad knowledge and inspirations.

Special thanks to my family and members of educational organizations that I have been working, for their endless motivation, constant mental support and love which have been influential in whatever I have achieved so far.

I wish to thank to all my colleagues and friends especially Mr. Deep Raj Sharma, and Mr. Ranjit Barma for supporting me directly and indirectly in this research work.
I have done my best to complete this research work. Suggestions from the readers are always welcomed, which will improve this work.

Nawa Raj Rimal
November, 2017

Abstract

Cloud computing is a popular computing concept that performs processing of huge volume of data using highly accessible geographically distributed resources that can be access by user on the basis of Pay per Use policy. In the modern computing environment where the amount of data to be processed is increasing day by day, the costs involved in the transmission and execution of such amount of data is mounting significantly. So there is a requirement of appropriate scheduling of tasks which will help to manage the escalating costs of data intensive applications. Complexity class of the task scheduling problem belongs to NP-complete involving extremely large search space with correspondingly large number of potential solutions and takes much longer time to find the optimal solution. There is no readymade and well-defined methodology to solve the problems under such circumstances. However, in cloud, it is sufficient to find near optimal solution, preferably in a short period of time.

The Evolutionary and Swarm based Algorithm is a scheduling algorithm capable of locating good solutions efficiently. The algorithm could be considered as belonging to the category of "Intelligent Optimization". This dissertation work analyzes Genetic Algorithm, Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) task scheduling algorithms in cloud computing that address the above-mentioned problem. The simulation result shows that the PSO algorithm produces the significant result and provides the better solution. General GA, ACO and PSO are implemented and tested. Evaluation results have shown that PSO seemed to be 38% better than GA and 26.5% better than ACO.

Keywords: Task Scheduling, Cloud Computing, Genetic Algorithm, Ant Colony Optimization, Particle Swarm Optimization

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACO	-	Ant Colony Optimization
ASP	-	Application Service Provision
BCO	-	Bee Colony Optimization
DPS	-	Dynamic Priority Scheduling
GA	-	Genetic Algorithm
ICT	-	Information, Communication and Technology
MGA1	-	Modified Genetic Algorithm 1
MGA2	-	Modified Genetic Algorithm 2
PSO	-	Particle Swarm Optimization
TSP	-	Travelling Salesman Problem

CHAPTER 1
INTRODUCTION AND BACKGROUND

1.1 Introduction to the Cloud Computing

Cloud Computing is a buzzword of 2010 and many experts disagree on its exact definition. But the most used one and concurred one includes the notion of web- based services which are available on demand from and optimized and highly scalable service provider. Since such a disagreement on the definition, one will be provided to better understand of the notion. The cloud is IT as a service, delivered by IT resources that are independent of location. It is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet where end- users have no knowledge of, expertise in, or control over the technology infrastructure (the cloud) that supports them [1].

Cloud computing is developed from the grid computing, distributed computing and parallel computing. It is a new business computing model and will distribute the tasks on the resources pool of large number computers, storage devices. Parallel and distributed system consisting of a collection of interconnected virtual computers. That presented as one or more unified computing resources based on service-level agreements between the service provider and consumers. There are three types of services of cloud computing [2]:

a. Software as a Service (SaaS)
b. Platform as a Service (PaaS)
c. Infrastructure as a Service (IaaS)

Platform as a Service (PaaS): PaaS is a deployment and development platform for applications provided as a service to developers over the Web. Third party renders develop and deploy software or applications to the end users through internet and servers. The cost and complexity of development and deployment of applications can be reduced to a great extent by developers by using this service. Thus, the developers can reduce the cost of buying and reduce the complexity of managing the required Infrastructure. It

provides all of the services required to build and deliver the web services to support the complete life cycle of web applications entirely from the Internet. This platform consists of infrastructure software, databases, middleware, and development tools.

Infrastructure as a Service (IaaS): is a delivery model associated with Hardware and Software as a service. Hardware such as Storage, server and network along with supporting software such as operating system, virtualization technology and file system. It is an 3 evolution of traditional hosting to allow users to provide resources on demand and without require any long term commitment. Different from PaaS services, the IaaS provider does very little management of data other than to keep the data center operational. Deployment and managing of the software services must be done by the end users just as the way they would in their own data center.

Software as a service (SaaS): SaaS allows access to programs to large number of users all the way through browser. For a user, this can save some cost on software and servers. For Service provider's, they only need to maintain one program, this can also save space and cost. Naturally, a SaaS provider gives access to applications to multiple clients and users over web by hosting and managing the given application in their or leased datacenters. SaaS providers also run their applications on platforms and infrastructure provided by other cloud providers.

1.1.1 Attributes of Cloud Computing

Before some of the attributes will be defined, the term cloud should be explained. A cloud has been long used in IT, in network diagrams respectively, to represent a sort of black box where the interfaces are well known but the internal routing and processing is not visible to the network users. Key attributes in cloud computing:

Service- Based: Consumer concerns are abstracted from provider concerns through service interfaces that are well- defined. The interfaces hide the implementation details and enable a completely automated response by the service provider. The service could be considered "ready to use" or "off the shelf" because it is designed to serve the specific

needs of a set of consumers, and the technologies are tailored to that need rather than the service being tailored to how the technology works. The articulation of the service feature is based on service levels and IT outcomes such as availability, response time, performance versus price, and clear and predefined operational processes, rather than technology and its capabilities. In other words, what the service needs to do is more important than how the technologies are used to implement the solution [3].

Scalable and Elastic: The service can scale capacity up or down as the consumer demands at the speed of full automation (from seconds for some services to hours for others). Elasticity is a trait of shared pools of resources. Scalability is a feature of the underlying infrastructure and software platforms. Elasticity is associated with not only scale but also an economic model that enables scaling in both directions in an automated fashion. This means that services scale on demand to add or remove resources as needed.

Shared: Services share a pool of resources to build economies of scale and IT resources are used with maximum efficiency. The underlying infrastructure, software or platforms are shared among the consumers of the service (usually unknown to the consumers). This enables unused resources to serve multiple needs for multiple consumers, all working at the same time.

Metered by Use: Services are tracked with usage metrics to enable multiple payment models. The service provider has a usage accounting model for measuring the use of the services, which could then be used to create different pricing plans and models. These may include pay- as- you go plans, subscriptions, fixed plans and even free plans. The implied payment plans will be based on usage, not on the cost of the equipment. These plans are based on the amount of the service used by the consumers, which may be in terms of hours, data transfers or other use- based attributes delivered.

Uses Internet Technologies: The service is delivered using Internet identifiers, formats and protocols, such as URLs, HTTP, IP and representational state transfer Web- oriented architecture. Many examples of Web technology exist as the foundation for

Internet- based services. Google's Gmail, Amazon.com's book buying, eBay's auctions sharing all exhibit the use of Internet and Web technologies and protocols. More details about examples are in the chapter four – Integration [3].

1.1.2 Cloud Computing Architecture

NIST (National Institute of Standards and Technology) is a well accepted institution all over the world for their work in the field of Information Technology. The working definition provided by NIST of Cloud Computing is provided here. NIST defines the Cloud Computing architecture by describing five essential characteristics, three cloud services models and four cloud deployment models.

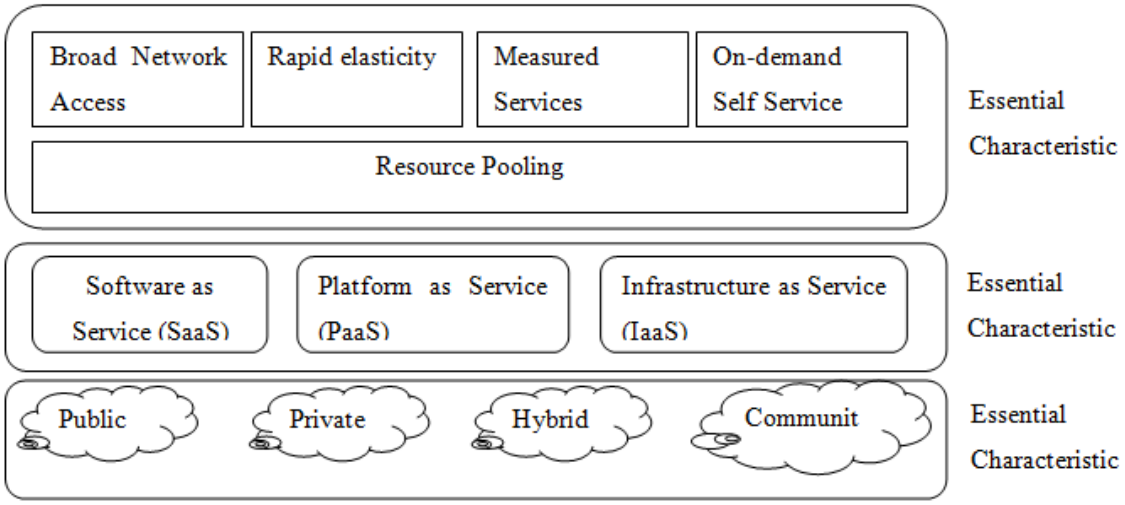

Figure1: Cloud definition provided by NIST

Similar to type of service, cloud may be hosted and deployed in different fashions depending on the use case. Cloud deployment models are as follows:

Private cloud: In private cloud model, the cloud infrastructure is deployed merely for single organization.

Community cloud: In community cloud model, the cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns.

Public cloud: In public cloud model, the cloud infrastructure is made available to public or a large industry group. The cloud is owned by an organization that sells service.

Hybrid cloud: In hybrid model, the cloud infrastructure is a composition of two or more clouds (private, community, or public).

1.2.3 History

History of Cloud Computing surprisingly began almost 50 years ago. The father of this idea is considered to be John McCarthy, a professor at MIT University in US, who first in 1961 presented the idea of sharing the same computer technology as being the same as for example sharing electricity. Electrical power needs many households/firms that possess a variety of electrical appliances but do not possess power plant. One power plant serves many customers and using the electricity example, power plant=service provider, distribution network=internet and the households/firms=computers [4].

Since that time, Cloud computing has evolved through a number of phases which include grid and utility computing, application service provision (ASP), and Software as a Service (SaaS). One of the first milestones was the arrival of Salesforce.com in 1999, which pioneered the concept of delivering enterprise applications via a simple website. The next development was Amazon Web Services in 2002, which provided a suite of cloud- based services including storage, computation and even human intelligence. Another big milestone came in 2009 as Google and others started to offer browser- based enterprise applications, though services such as Google Apps [5].

1.2 Task and Task Scheduling in Cloud system

Task scheduling is an important and challenging issue of cloud computing environment. A task is a sequential activity that uses a set of inputs to produce a set of outputs. Every application is completely different and is independent but the task with in application

may be dependent with each other. For example, some require more CPU time to compute complex task, and some others may need more memory to store data, etc.

Its main purpose is to schedule tasks to the adaptable resources in accordance with adaptable time, which involves finding out a proper sequence in which all the tasks can be executed such that execution time and execution cost can be minimized.

1.2.1 Genetic Algorithm

The concept of genetic algorithm was developed by Holland and his colleagues in the 1970s [6]. Genetic Algorithm (GA) inspired by Darwin's theory of evolution. It was the first evolutionary technique that is based on the principle of natural selection. 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.

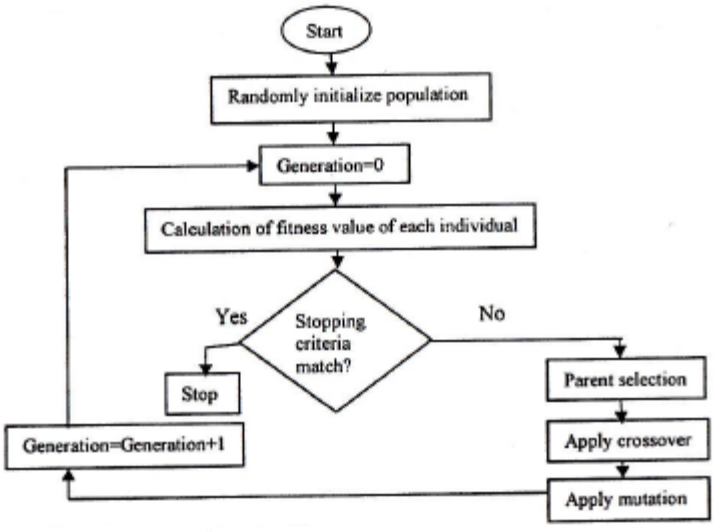

Figure 2: Flow Chart of Genetic Algorithm

Basic Steps in Genetic Algorithm

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.

Encoding and Representation

In GA terminology, a solution vector is called an individual or a chromosome. Chromosomes are made of discrete units called genes. Many data structures such as array, lists or trees are used in many ways for representing these chromosomes. The length of such data structures is equal to the number of tasks and resources represent their elements.

Population Initialization

The initial chromosome is used to generate the initial population. Initial population generator function takes the initial chromosome as input to generate new chromosomes which form population of initial generation. To design a new chromosome the gene values of chromosome are taken as the key elements. The digits are randomly arranged in the positions of the new chromosome.

Fitness Function and Selection

The fitness function generates a fitness value of each chromosome. The value indicates how suitable for solving a task scheduling problem a chromosome is. The fitness function for general task scheduling problems is based on execution time of tasks. But, task scheduling problems in Cloud computing are different from general task scheduling problems because computing services in Cloud computing are offered through a Service level argument (SLA) between cloud users and providers. The minimization of execution time is a goal of task scheduling in Cloud computing. Therefore, the fitness function in our genetic algorithm is defined by these elements.

Crossover and Mutation

The crossover operation is a simple mechanism to swap one part of a chromosome for that of another chromosome. The one-point crossover is used for transferring genetic material of parent to children.

The mutation operation is to expand the search space by changing one part of a chromosome. In the beginning of the genetic algorithm, the quality of generated chromosomes is not particularly good. As time goes by, the quality of chromosomes becomes more improved. There is the potential for improvement in quality by the mutation operation in the early part of the genetic algorithm.

Restart & Stop Condition

If the best fitness value of the current population does not reach the minimum fitness threshold, chromosomes with high fitness values in the current population fill the half of the next population. And, the rest of the next population is filled with chromosomes generated randomly.

1.2.2 Ant Colony Optimization

The ant colony optimization meta-heuristic (ACO) was first described by Marco Dorigo in 1992 in his PhD thesis, and was initially applied to the travelling salesman problem (TSP). Many species of ant deposit a chemical *pheromone* trail as they move about their environment, and they are also able to detect and follow pheromone trails that they may encounter [7]. Interaction among ants and the environment is based on pheromones. Ants can smell the pheromone and tend to choose paths marked by strong pheromone concentrations.

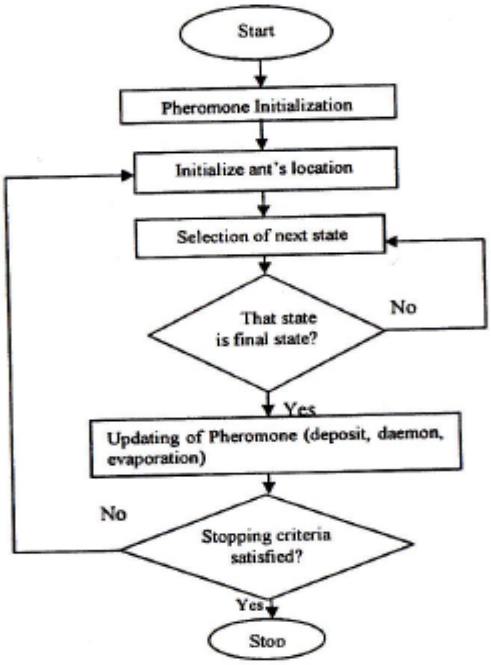

Figure 3:Flow Chart of ACO

Basic Steps in Ant Colony Optimization

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 if it not then repeat 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 the stop the execution, else repeat from step 2.

Probability matrix

The probability of picking a resource i to which task j will be allocated, $Prob_{ij}$ is determined as following.

$$\text{Prob}(\text{i},\text{j}) = \frac{(\tau(i,\text{j})^{\alpha} * [\eta(i,\text{j})]^{\beta})}{\sum_{s \in p} (\tau(i,s)^{\alpha} * [\eta(i,s)]^{\beta})}$$

whereτ(i,j)and η(i, j) are the pheromone value and heuristic value associated with the component and α, β is a parameter used to define the extent to which the ants use the pheromone information and heuristic information.

1.2.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a self-adaptive global search based optimization technique introduced by Kennedy and Eberhart [8]. The algorithm is similar to other population-based algorithms like Genetic algorithms but, there is no direct re-combination of individuals of the population.

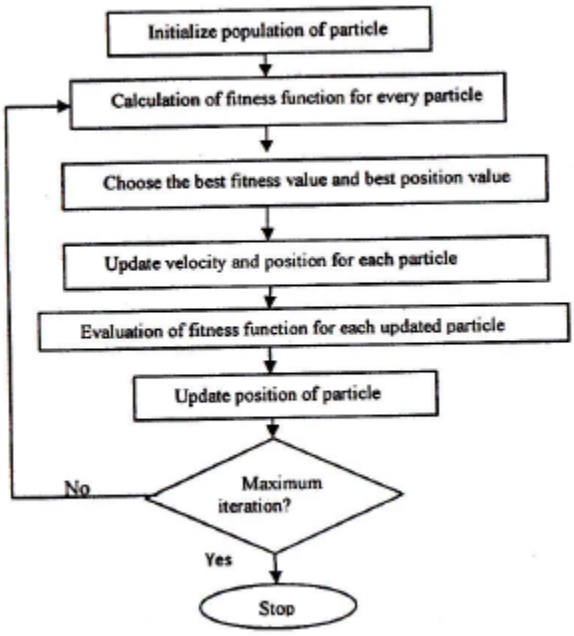

Figure 4: Flow Chart of PSO

Basic steps in Particle Swarm Optimization

1. Population is initialized with particle of random position and velocities.
2. The fitness function is calculated for every particle.
3. If the fitness value is better than the previous best pbest, set current fitness value as the new pbest.

4. After step 2 and step 3 for all particles select best particle as gbest.
5. By using the following (1) and (2) equation, particle's position and velocity will update.
6. If the number of iteration is reached maximum, then stop otherwise repeat from step 2.

The velocity and position of the particle can be calculated by

$$v_i^{k+1} = wv_i^K + c_1r_1(pbest_i - x_i^k) + c_2r_2(gbest_i - x_i^k) \quad (1)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} (2)$$

where:

v_i^k	velocity of particle i at iteration k
v_i^{k+1}	velocity of particle i at iteration k+1
w	intertia weight
c_j	acceleration coefficients, j=1,2
r_i	random number between 0 and 1; i=1,2
x_i^k	current position of particle i at iteration k
$pbest_i$	best position of particle i
$gbest_i$	position of best particle in a population
x_i^{k+1}	position of the particle i at iteration k+1.

1.3 Problem Statement

Task scheduling is one of the most important and critical problems in cloud computing. The problem of task scheduling is a NP-complete problem. Task scheduling problem is defined as the problem of scheduling a set T of n independent task $T_i = \{t_1, ..., t_n\}$, on a set R of m resources, $R_j = \{r_1, ..., r_m\}$ such that all the jobs are completed as quickly as possible, i.e. minimizing the makespan of the solution. Makespan is equal to time that execution of last task on processing resources is finished. Suppose that execution time of task i on computational resources j is known, and equal to jobRunTime(ij).

Makespan(MS)= max $(\sum x_{i,j})$ where, $i \in T$ and $j \in R$. (1)

Find the schedule that Minimize Total Cost of executing the task on the available resource with satisfying the following constraints.

a) At any instance, a task is executing on only one resource.
b) Tasks are independent to each other.
c) Ensure that a task hosted on only on a resource which is operating.
d) Total processing requirements of all tasks hosted on resource should not exceed the maximum processing capacity of that resource.
e) Total memory requirements of all tasks hosted on resource should not exceed the maximum memory available with that resource.

1.4 Objective

The objective of this work in particular is to design efficient task scheduling algorithm and other general objectives are

1) To compare the different variation of GA, ACO and PSO algorithm to solve task scheduling in cloud.
2) To implementation of GA, ACO and PSO in cloud framework and find the best algorithm from these three.

1.5 Thesis organization

The rest of this thesis is organized as: chapter2 is a survey of the major existing task scheduler in cloud computing environment, chapter 3 details the research methodology of the Genetic algorithms (GA), Ant colony optimization (ACO) and Particle swarm optimization (PSO) based task scheduling problem for cloud environment, chapter 4 presents the experiment and result, and chapter 5 concludes the thesis, summarizing its achievements and further recommendations.

CHAPTER 2
REVIEW OF THE LITERATURE

Cloud computing is an emerging fields of information technology which aim to provides information, communication and technology (ICT) services as a utility. It will be beneficial for both ICT industry and customer if this technology get success. It is necessary to improve the utilization of data center to provide these services on demand. To improve the utilization of data center resource, the scheduling of task in to a set of data center plays an important role. If the schedule of task assignment into a set of data center is optimal then it produces the efficient use of resource and increase the performance of system. In the data center, hundreds of servers run thousands of tasks that come and keep receiving so many requests are in queue. In this context, one needs to find few target servers out of many powered servers, which can fulfill a batch of incoming tasks. So, Task scheduling is an important issue which is greatly influences the performance of cloud service provider. The task scheduling problem belongs to NP-complete complexity class, involving extremely large search space with correspondingly large number of potential solutions and takes much longer time to find the optimal solution. There is no readymade and well-defined methodology to solve the problems under such circumstances. However, in cloud, it is sufficient to find near optimal solution, preferably in a short period of time.

2.1 Previous Work

In the work presented in [9], the genetic algorithm is developed to solve the task scheduling with near optimal solution and two modifications on basic genetic algorithm are also applied to get better result. The simulation result shows that the modification on basic genetic algorithm produces the significant result and provides the better solution.

The genetic algorithm is simple to implement but it is powerful to solve complex problem such as scheduling of task on multiple machine. The final schedule produced by GA depends upon the initial population used. If the initial population provided to GA, is more near to solution, it produces better solution with less time. The results show that MGA2 is

slightly better than MGA1 in terms of total Make-span. In some experiment, the MGA1 has produced minimum make-span. “It is due to the random chromosome included in the initial population of MGA1 and MGA2”. In conclusion, from experimentation in[9], the MGA2 is better than MGA1.

The different data set were designed with the expected output as a minimum makespan time. The analysis of output of GA, ACO and MMAS algorithms was done and which one produced minimum makespan for scheduling was analyzed in [10].

2.2 Genetic Algorithm

Genetic Algorithm strategy is one of the heuristic strategies. By using it we search to get a near-optimal solution in many large solution spaces. At first, we randomly initialize a population of chromosomes which we generally consider as possible scheduling for a given particular task. We can get a fitness value (make span) which is acquired from the scheduling of tasks to particular machines within that chromosome. After getting the initial population, we will evaluate all the chromosomes in the population based on their respective fitness value and we consider the smaller make span as a better mapping. We perform Selection scheme which duplicates some chromosomes probabilistically and deletes others and at the same time we can find that better mappings have more probability of being duplicated in successive generation and we will find that population size remains constant in all the generations. Next, we perform a crossover operation which select a random pair of chromosomes of a task and picks a random point in first chromosome. This operation also exchanges machine assignments between particular corresponding tasks. Next, we perform mutation operation after crossover which randomly selects a chromosome and then selects a task randomly within the chromosome and finally reassigns randomly it to a new machine. We start another iteration of Genetic Algorithm after evaluating the new population and there after we perform all the above-mentioned operations like selection, crossover, mutation and evaluation. We just stop the iteration when the stopping criteria are met.

Basic Steps in Genetic Algorithm

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.

Savitha P et. al. (2013)[11] proposed a genetic algorithm (GA) for task scheduling in cloud computing environment. This proposed algorithm is well tested and results are compared with the existing genetic algorithm based workflow scheduling techniques. The results of proposed GA outperform the system's throughput.

2.3 Ant Colony Optimization

Ant Colony Optimization (ACO) is a type of swarm which carries out the work based on the indirect communication. This type of communication can be seen among the real ants that use a substance called pheromone in order to communicate. Some artificial ants while conveying about their particular experience regarding solving a problem they act as special agents try to use numerical information which is also called as artificial pheromone information. All these fundamentals are going to provide a basic framework to several combinational problems of optimization in applications of ant algorithms. Hence, we call the algorithms as ACO algorithms if they were acquired from ACO swarm.

In order to design swarm algorithms, we mainly consider Ant Colony Optimization (ACO) since they can solve many combinational optimization problems. The first or earliest algorithm which we categorized in this framework was presented in 1991and, since then several modifications regarding the principles were documented in the literature. The combination of both priori and posterior information is the main characteristic of ACO algorithms. We can have the basic structure or design of promising or best solution in priori information while we can have the design or structure of earlier

obtained good solutions in posterior information. Algorithms like Meta heuristic algorithms work up on some basic heuristic with the main intention to get rid of local optima. The basic heuristic may be constructive heuristic which starts from null solution and finally get a complete solution by adding elements or it can be a local heuristic where you have a complete solution and finally you acquire a better solution by iteratively modifying some elements in it. But the Meta heuristic part allows the low-level heuristic to attain good solutions since you cannot get them from the heuristic when it is iterated. We can have control over these heuristics either by randomizing or constraining or you can control by mixing the elements which you derived from various different solutions.

Basic Steps in Ant Colony Optimization

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 if it not then repeat 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 the stop the execution, else repeat from step 2.

C.W. Chiang et. al. (2006)[12] has proposed the task scheduling and matching using ant colony optimization. Here one algorithm is proposed named ACO-TMS that reduces the scheduling time and always help to search a satisfactory scheduling result by integrating local search procedure. The proposed technique is compare with some existing approach like GA and DPS heuristic. The new technique gives better result and minimizes the time as compared to existing methodologies.

Linan Zhu et. al. (2012)[13] proposed ant colony optimization techniques to overcome the problem related to scheduling of resources. Here basic ant colony optimization is used to analyze and design the scheduling of resources in cloud. CloudSim simulation tool is used to balance and distribute of load of nodes for better performance.

2.3.1 ACO algorithm for TSP

ACO is a probabilistic technique where searching is done for the optimal path in the graph based on behavior of ants seeking a path between their colony and source of food. Algorithm for TSP is presented below:

- Often applied to TSP: shortest path between n nodes
- Algorithm in Pseudocode:
    1. **Initialize** Trail
    2. **Do While** (Stopping Criteria Not Satisfied) – Cycle Loop
        2.1 Do Until (Each Ant Completes a Tour) – Tour Loop
        2.2 Local Trail Update
        2.3 End Do
        2.4 Analyze Tours
        2.5 Global Trail Update
    3. **End Do**

2.3.2 ACO in job scheduling problems

In the paper [14], they apply the collective intelligence of many simple agents to the problem of Job Shop Scheduling, which consists of finding an optimal plan that minimizes the makespan, which is the time required to perform a finite number of tasks in a finite number of machines. Each task is a sequence of operations, each one with a determined machine and processing time. Feasible solutions must comply with the restrictions that apply to the problem of Job Shop Scheduling, as respecting the precedence between operations determining the technological sequence without interrupting any operations until completion. The operations conform the graph nodes that represent the problem, united by edges in which ants are moving. Each individual only has local information of the system that shares through a hormone called pheromone.

2.4 Particle Swarm Optimization Algorithms

Particle Swarm Optimization (PSO) is an evolutionary approach to optimization technique. It is a population based stochastic technique inspired by bird flocking and fish

schooling. First developed by James Eberhart and Kennedy in 1995, the algorithm maintains swarm of particles in multidimensional space. Each particle in the swarm is a potential solution to the given problem [8]. Each particle moves through a problem search space seeking an optimal or better-found solution; broadcasts its current position to neighboring particles seeking an optimal or better found solution.

Basic steps in Particle Swarm Optimization

1. Population is initialized with particle of random position and velocities.
2. The fitness function is calculated for every particle.
3. If the fitness value is better than the previous best pbest, set current fitness value as the new pbest.
4. After step 2 and step 3 for all particles select best particle as gbest.
5. By using the above (1) and (2) equation, particle's position and velocity will update.
6. If the number of iteration is reached maximum, then stop otherwise repeat from step 2.

Suraj Pandey et. al. (2010)[15] proposed heuristic based particle swarm optimization for task scheduling to optimize the cost associated with computation and communication. PSO can properly balance the workflow and saves the cost as compared to existing technique.

CHAPTER 3
RSEARCH METHODOLOGY

In this chapter, the outline of the proposed work is discussed. The first few paragraphs in this chapter describe the algorithms used to solve the task scheduling problem. The remaining of the chapter explains the details of Genetic ACO and PSO algorithm implemented in this work.

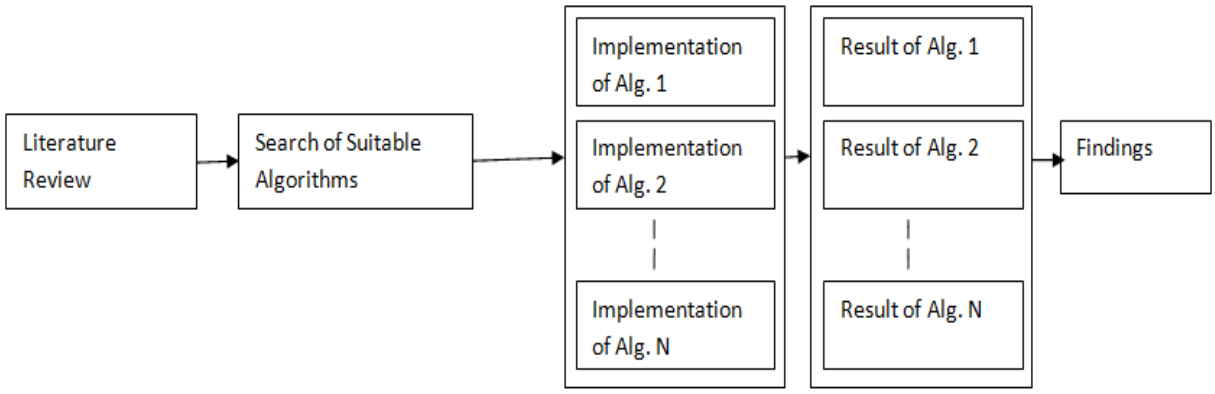

Figure 5: Framework of Research

3.1 Search of Suitable Algorithm

From the literature, a conclusion came that there are lot of algorithms for Task scheduling optimization problem. From them, we selected GA, ACO, and PSO for our work.

3.2 Algorithms

GA, ACO and PSO are modified in some extent without spoiling their taste to solve the scheduling problem in our scenario. Our Problem and the modified algorithms to solve the problem are presented below:

3.3 The Problem and modified Algorithms for the problem

We have assumed a scenario where processing power of every Machine (Resource) is same but with different holding Capacity (RAM capacity). Each Task (Process) has two

properties associated with it. The first property amount of RAM space it would take. This space is termed as 'power' of the Task. And the second property is the amount of time required inside the Machine (Resource) to finish the task.

3.3.1 Modified Genetics Algorithm

The modified algorithm for solving this problem is presented here:

Given:

a. ***Number of Resources, a set R = { R_1 ,R_2,R_3…… R_N } ,***
holding capacity, a set C = {c_1,c_2,c_3 ,…… cN}

b. ***Number of Tasks, a set T= {t_1,t_2,t_3……..t_N}, two lists associated with T;***
First list is the power and the second list is time needed to complete the task in RAM.

Procedure:

1. Initialize the Resources with 'legal' Tasks.
2. While the Task list is not empty:
    2.1 Calculate the available capacity of all the Resources
    2.2. Create all the possible combinations for 'Excluding' or 'Including' of a Resource with respect to 'legal' Task
    2.3 Let I= {I_1, I_2, I_3, ….}be the set of Included task.
    2.4 Calculate the weight of I defined as follows:
        Weight= 0
        For element in I:
            Weight= Weight+ Available Resource of element
        Return weight
    2.5 Find the best I
    2.6 Schedule jobs to elements of best I
    2.7 Update task remaining and completed in elements of I

The 'Legal' task mentioned in the above algorithm are the tasks having power <=the available capacity (RAM) of the Resource.

3.3.2 Modified Ant Colony Algorithm

Given:

a. ***Number of Resources, a set R = { R_1 ,R_2,R_3…… R_N } ,***
Holding capacity, a set C = {c_1,c_2,c_3 ,…… c_N}

b. ***Number of Tasks, a set T= {t_1,t_2,t_3……..t_N}, two lists associated with T;***
First list is the power and the second list is time needed to complete the task in RAM.

Procedure:

1. Initialize the Resources with 'Legal' Tasks
2. While the Task list is not empty:
    a. Calculate visibility matrix
    b. Calculate Pheromone list
    c. Calculate Probability list
    d. On the basis of Probability, Choose the Task from task list to send on applicable Resource
    e. Update Task remaining and Completed in elements of I

3.3.3 Modified Particle Swarm Optimization

Given:

a. ***Number of Resources, a set R = { R_1 ,R_2,R_3…… R_N } ,***
Holding capacity, a set C = {c_1,c_2,c_3 ,…… c_N}

b. ***Number of Tasks, a set T= {t_1,t_2,t_3……..t_N}, two lists associated with T;***
First list is the power and the second list is time needed to complete the task in RAM.

Procedure:

1. Initialize the Resources with 'Legal' Tasks
2. While the Task list is not empty:
    a. Broadcast the available Capacity by each Resource
    b. Select the 'Legal' Task and assignment of that task to the fittest Resource

c. Update Task remaining and Completed in elements of I

3.4 Implementation of Algorithms

In order to understand how to implement an Algorithm, we first need to conceptually understand what an Algorithm is. An Algorithm is a series of steps that we expect will arrive at a specific solution. Writing a program does not equal expressing code, that idea ignores and neglects the entire idea of writing code to solve a problem.

1. Establish the Rules of a Problem: A problem can be interpreted in many ways. The outcome of the problem is how we respond to those problems.
2. We need to specify a Plan that Should Solve the Problem and Elaborate the Plan into Steps: Come up with a sequence of steps that we can explain to a very obedient preschooler. There should be no room for confusion for what these steps really mean.
3. Translate each step into a line of code: This step is deterministic. The process of translating from one definitely clear solution in any natural language can be easily translated to a correct programming language.

The implementation of the algorithms was done within the following Resources:

a. Core I5 Intel Processor with 4 GB RAM
b. Compaq Dual Core processor with 2 GB RAM

The Programming languages used were:

a. JAVA 1.8
b. PYTHON 2.7

3.5 Analyzing Results

After completion of implementing Algorithms, the Analysis of Result was done which is explained in the upcoming chapter. Basically, the conclusion of our results is based totally on experiments. Almost 20 Datasets were generated randomly, using a Computer

program to give the input for the three Algorithms. The Computer program used for preparation of data set is shown below:

```
import random
import time
from threading import Thread
time_list = [1,2,3,4,5,6,7,8,9,10]
capacity_list = [10,11,12,13,14,15,16,17,18,19,20,21,22]
power_list = [5,6,7,8,9,10]
total_resources = int(raw_input("Enter number of Resources:"))
total_tasks=int(raw_input("Enter total number of task: "))
res_list = []
task_list = []
fori in range(total_resources):
        res_list.append((random.choice(capacity_list),[]))
fori in range(total_tasks):
                task_list.append((random.choice(power_list),random.choice(time_list)))
print "Task List:"
printtask_list
print "Resource List:"
printres_list
```

CHAPTER 4
RESULT, ANALYSIS AND COMPARISIONS

4.1 Experimental Setup

The experimental environment consists of Intel Core™ i5-2450 2.5 GHz 4G RAM configuration, and window 10 Pro, Ubuntu 16.04 LTS Operating System. The simulation idea as shown in figure 5.1 is implemented on the machine with Eclipse IDE with java SDK 1.8 and Python 2.7. The class resource allocator consists of information such as status of resource, capacity of resources and processing power of resources. Similarly the class Task to represent the set of task also consist of information such as the memory requirement, burst time of given task, status of given task either finish or running or interrupted. Scheduler Algorithm is invoked accordingly. Each task is removed from set T after it is served.

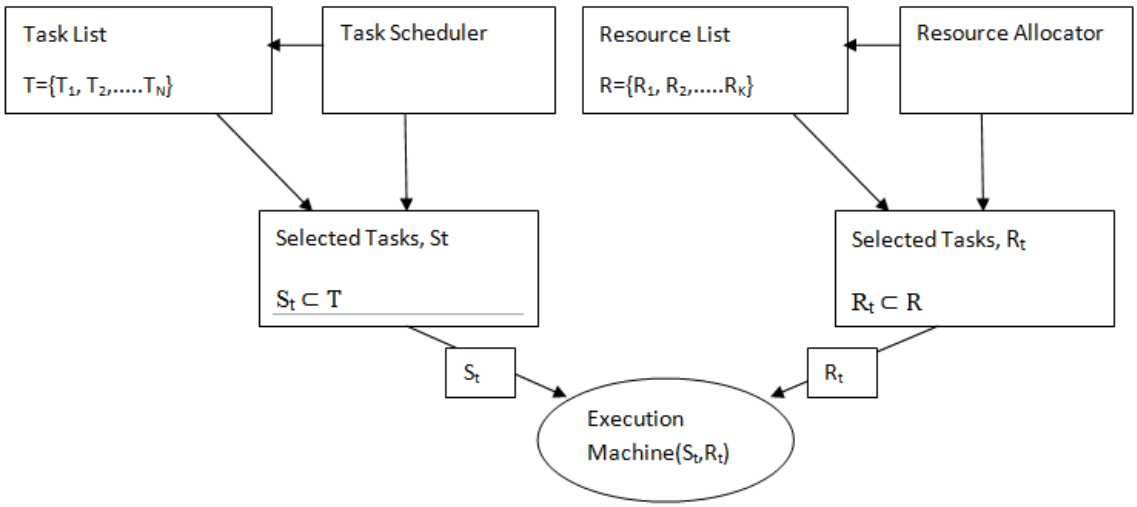

Figure 6: System Flowchart for Task Scheduling

4.2 Generation of Data set

Task list is generated randomly and it is added to the set T. Arrival time of the task is recorded or assumed to be zero. The total numbers of tasks were set 7 to 35, the

processing times of tasks are uniformly distributed in [1, 15] and the memory requirement for each task is also uniformly distributed in [7, 50]. The numbers of Data Centers are from 2 to 20. In all experiments of the following sections the numeric parameters α=1 and β=2 values are taken [13].

4.3 Trace of Algorithms

Here datacenters are represented by R = {R_1, R_2….., RN} assume as resources and tasks are represented by T = {T_1, T_2, ……, TN}.

The capacity of resources is given in braces like R_1(12) that means resource R_1 has computing capacity 12. The power and time required to execute task are given in braces like T_1(6,3) where 6 is power of task and 3 is time required to execute task on any resources.

To trace all three algorithms, we will take five resources and 13 tasks to schedule on those resources.

4.3.1 Trace of GA

Resources: R_1(14), R_2(18), R_3(10), R_4(12), R_5(15)

Tasks: T_1(7,6), T_2(9,7), T_3(5,7), T_4(7,3), T_5(7,1), T_6(8,2), T_7(10,6), T_8(6,6), T_9(10,5), T_{10}(8,2), T_{11}(7,3), T_{12}(7,2), T_{13}(6,7)

Time 0

Initialization: Randomly pick the 'legal' tasks for processing in Resources

#X# where X represents available space of the resources.

R_1(14) →T_2(9,7) #5#

R_2(18) →T_{12}(7,2) #11#

R_3(10) →T_9(10,5) #0#

R_4(12) →T_2(5,7) #7#

R_5(15) →T_{10}(8,2) #7#

Chromosome generation:

	Chromosome	Weight
1.	00000	0
2.	00001	7
3.	00010	7
4.	00011	14
5.	00100	0
6.	00101	7
7.	00110	7
8.	00111	14
9.	01000	11
10.	01001	18
11.	01010	18
12.	01011	25
13.	01100	11
14.	01101	18
15.	01110	18
16.	01111	25
17.	10000	5
18.	10001	12
19.	10010	12
20.	10011	19
21.	10100	5
22.	10101	12
23.	10110	12
24.	10111	19
25.	11000	16
26.	11001	23
27.	11010	23
28.	11011	30
29.	11100	16
30.	11101	23

31. 11110 23

32. 11111 30

Selection of fittest chromosome: 01111(25) and 11011(30)

Crossover Result 11011, 01111

Mutation Result 01011(25), 11111(30)

Therefore, new task will be scheduled to R_1, R_2, R_3, R_4, R_5

$R_1(14) \rightarrow T_2(9,7)$ #5#

$R_2(18) \rightarrow T_{12}(7,2), T_{11}(7,3)$ #4#

$R_3(10) \rightarrow T_9(10,5)$ #0#

$R_4(12) \rightarrow T_2(5,7), T_8(6,6)$ #1#

$R_5(15) \rightarrow T_{10}(8,2), T_{13}(6,7)$ #1#

Time 1

$R_1(14) \rightarrow T_2(9,6)$ #5#

$R_2(18) \rightarrow T_{12}(7,1), T_{11}(7,2)$ #1#

$R_3(10) \rightarrow T_9(10,4)$ #0#

$R_4(12) \rightarrow T_2(5,6), T_8(6,5)$ #1#

$R_5(15) \rightarrow T_{10}(8,1), T_{13}(6,6)$ #1#

Time 2

$R_1(14) \rightarrow T_2(9,5)$ #5#

$R_2(18) \rightarrow T_{12}(7, \text{completed}), T_{11}(7,1)$ #11#

$R_3(10) \rightarrow T_9(10,3)$ #0#

$R_4(12) \rightarrow T_2(5,5), T_8(6,4)$ #1#

$R_5(15) \rightarrow T_{10}(8, \text{completed}), T_{13}(6,5)$ #9#

Chromosome generation:

Chromosome	Weight
33. 00000	0
34. 00001	9
35. 00010	1
36. 00011	10
37. 00100	0
38. 00101	9
39. 00110	1
40. 00111	10
41. 01000	11
42. 01001	20
43. 01010	12
44. 01011	21
45. 01100	11
46. 01101	20
47. 01110	12
48. 01111	21
49. 10000	5
50. 10001	14
51. 10010	6
52. 10011	15
53. 10100	5
54. 10101	14
55. 10110	6
56. 10111	15
57. 11000	16
58. 11001	25
59. 11010	17
60. 11011	26
61. 11100	16

62. 11101	25
63. 11110	17
64. 11111	26

Selection of fittest chromosome: 11011(26) and 11001(25)

Crossover Result 11001, 11011

Mutation Result 11011(26), 11001(25)

Therefore, new task will be scheduled to R_1, R_2, R_3, R_4

$R_1(14) \rightarrow T_2(9,5)$	#5#
$R_2(18) \rightarrow T_{11}(7,1), T_7(10,6)$	#1#
$R_3(10) \rightarrow T_9(10,3)$	#0#
$R_4(12) \rightarrow T_2(5,5), T_8(6,4)$	#1#
$R_5(15) \rightarrow T_{13}(6,5), T_6(8,2)$	#1#

Time 3

$R_1(14) \rightarrow T_2(9,4)$	#5#
$R_2(18) \rightarrow T_{11}(7,$ completed$), T_7(10,5)$	#8#
$R_3(10) \rightarrow T_9(10,2)$	#0#
$R_4(12) \rightarrow T_2(5,4), T_8(6,3)$	#1#
$R_5(15) \rightarrow T_{13}(6,4), T_6(8,1)$	#1#

Chromosome generation:

Chromosome	Weight
65. 00000	0
66. 00001	1
67. 00010	1
68. 00011	2
69. 00100	0

70. 00101	1
71. 00110	1
72. 00111	2
73. 01000	8
74. 01001	9
75. 01010	9
76. 01011	10
77. 01100	8
78. 01101	9
79. 01110	9
80. 01111	10
81. 10000	5
82. 10001	6
83. 10010	6
84. 10011	7
85. 10100	5
86. 10101	6
87. 10110	6
88. 10111	7
89. 11000	13
90. 11001	14
91. 11010	14
92. 11011	15
93. 11100	13
94. 11101	14
95. 11110	14
96. 11111	15

Selection of fittest chromosome: 11011(15) and 11110(14)

Crossover Result 11111, 11010

Mutation Result 11110(14), 11011(15)

Therefore, new task will be scheduled to R_1, R_2, R_4, R_5

Similarly, Time 4, Time 5 and Time 6 were calculated and total time taken for all tasks to be completed = 12 units of time.

4.3.2 Trace of ACO

Resources: $R_1(14)$, $R_2(18)$, $R_3(10)$, $R_4(12)$, $R_5(15)$

Tasks: $T_1(7,6)$, $T_2(9,7)$, $T_3(5,7)$, $T_4(7,3)$, $T_5(7,1)$, $T_6(8,2)$, $T_7(10,6)$, $T_8(6,6)$, $T_9(10,5)$, $T_{10}(8,2)$, $T_{11}(7,3)$, $T_{12}(7,2)$, $T_{13}(6,7)$

Time 0

Initialization: Randomly pick the 'legal' tasks for processing in Resources

#X# where X represents available space of the resources.

$R_1(14) \rightarrow T_7(10,6)$ #4#

$R_2(18) \rightarrow T_{11}(7,3)$ #11#

$R_3(10) \rightarrow T_2(9,7)$ #1#

$R_4(12) \rightarrow T_{10}(8,2)$ #4#

$R_5(15) \rightarrow T_{12}(7,2)$ #8#

Calculation of Visibility matrix=

[

[0, 0.02040816326530612, 0.1111111111111111, 0, 0.0625],

[0.02040816326530612, 0, 0.010000000000000002, 0.02040816326530612, 0.1111111111111111],

[0.1111111111111111, 0.010000000000000002, 0, 0.1111111111111111, 0.02040816326530612],

[0, 0.02040816326530612, 0.1111111111111111, 0, 0.0625],

[0.0625, 0.1111111111111111, 0.02040816326530612, 0.0625, 0]

]

Pheromone calculation =

[

[0.2857142857142857],

[0.6111111111111112],
[0.1],
[0.3333333333333333],
[0.5333333333333333]
]

Probability Matrix calculation =
[
[0.0, 0.10244735344336936, 0.5577689243027888, 0.0, 0.31374501992031867],
[0.17109564913393016, 0.0, 0.0838368680756258, 0.17109564913393016, 0.9315207563958419],
[0.12953253733973652, 0.011657928360576289, 0.0, 0.12953253733973652, 0.023791690531788336],
[0.0, 0.11952191235059759, 0.6507304116865869, 0.0, 0.36603585657370513],
[0.30684486707447894, 0.5455019859101847, 0.10019424231003393, 0.30684486707447894, 0.0]
]

Since max (probability matrix) is at 2nd Row and 5th column; therefore, New Tasks will be scheduled in these Resources R_2 and R_5.

	Time 1
$R_1(14) \rightarrow T_7(10,6)$	#4#
$R_2(18) \rightarrow T_{11}(7,3), T_9(10,4)$	#1#
$R_3(10) \rightarrow T_2(9,7)$	#1#
$R_4(12) \rightarrow T_{10}(8,2)$	#4#
$R_5(15) \rightarrow T_{12}(7,2), T_5(7, \text{completed}), T_6(8,2)$	#0#

Visibility matrix calculation =

[

[0, 0.1111111111111111, 0.1111111111111111, 0, 0.0625],

[0.1111111111111111, 0, 0, 0.1111111111111111, 0.02040816326530612],

[0.1111111111111111, 0, 0, 0.1111111111111111, 0.02040816326530612],

[0, 0.1111111111111111, 0.1111111111111111, 0, 0.0625],

[0.0625, 0.02040816326530612, 0.02040816326530612, 0.0625, 0]

]

Pheromone calculation =

[

[0.2857142857142857],

[0.05555555555555555],

[0.1],

[0.3333333333333333],

[0.5333333333333333]

]

Probability matrix calculation =

[

[0.0, 0.627177700348432, 0.627177700348432, 0.0, 0.352787456445993],

[0.07748260594560405, 0.0, 0.0, 0.07748260594560405, 0.014231499051233394],

[0.13946869070208728, 0.0, 0.0, 0.13946869070208728, 0.025616698292220113],

[0.0, 0.7317073170731707, 0.7317073170731707, 0.0, 0.4115853658536585],

[0.7962085308056872, 0.25998645903859174, 0.25998645903859174, 0.7962085308056872, 0.0]

]

Since max (probability matrix) is at 5th Row and 1st column; therefore, New Tasks will be scheduled in these Resources R_5 and R_1.

Similarly, Time 2, Time 3, Time 4 were calculated and total time taken for all tasks to be completed = 10 units of time.

4.3.3 Trace of PSO

Resources: $R_1(14)$, $R_2(18)$, $R_3(10)$, $R_4(12)$, $R_5(15)$

Tasks: $T_1(7,6)$, $T_2(9,7)$, $T_3(5,7)$, $T_4(7,3)$, $T_5(7,1)$, $T_6(8,2)$, $T_7(10,6)$, $T_8(6,6)$, $T_9(10,5)$, $T_{10}(8,2)$, $T_{11}(7,3)$, $T_{12}(7,2)$, $T_{13}(6,7)$

Time 0

Initialization: Randomly pick the 'legal' tasks for processing in Resources

#X# where X represents available space of the resources.

$R_1(14) \rightarrow T_{12}(7,2)$ #7#

$R_2(18) \rightarrow T_2(9,7)$ #9#

$R_3(10) \rightarrow T_{13}(6,7)$ #4#

$R_4(12) \rightarrow T_9(10,5)$ #2#

$R_5(15) \rightarrow T_6(8,2)$ #8#

Broadcast available capacity from each resources and legal task will assign to resources.

$R_1(14) \rightarrow T_{12}(7,2), T_1(7,6)$ #0#

$R_2(18) \rightarrow T_2(9,7), T_{11}(7,3)$ #2#

$R_3(10) \rightarrow T_{13}(6,7)$ #4#

$R_4(12) \rightarrow T_9(10,5)$ #2#

$R_5(15) \rightarrow T_6(8,2), T_5(7,1)$ #0#

Time 1

$R_1(14) \rightarrow T_{12}(7,1), T_1(7,5)$ #0#

$R_2(18) \rightarrow T_2(9,6), T_{11}(7,2)$ #2#

$R_3(10) \rightarrow T_{13}(6,6)$ #4#

$R_4(12) \rightarrow T_9(10,4)$ #2#

$R_5(15) \rightarrow T_6(8,1), T_5(7,$ completed) #7#

Broadcast available capacity from each resources and legal task will assign to resources.

$R_1(14) \rightarrow T_{12}(7,1), T_1(7,5)$ #0#

$R_2(15) \rightarrow T_2(9,6), T_{11}(7,2)$ #2#

$R_3(10) \rightarrow T_{13}(6,6)$ #4#

$R_4(12) \rightarrow T_9(10,4)$ #2#

$R_5(15) \rightarrow T_6(8,1), T_{10}(7,2)$ #0#

Similarly, Time 2, Time 3, Time 4 were calculated and total time taken for all tasks to be completed = 9 units of time.

4.4 Results and Analysis

The results of the simulation are represented in graph. Simulation results demonstrate that more iterations or number of particles obtain the better solution since more solutions were generated.

4.4.1 Comparative Analysis of GA, ACO and PSO

Different number of Resources and Tasks were taken at random and for almost 90 % of the cases, PSO was seen best. The tabular and graphical comparisons of these three algorithms with three different scenarios are presented below.

Time	Tasks Completed		
	GA	ACO	PSO
0	0	0	0
1	0	0	3
2	1	0	3
3	3	1	4
4	4	4	8
5	6	8	11
6	7	9	13
7	9	10	-
8	10	13	-
9	11	-	-
10	13	-	-

Table 1 Resource 3 and Tasks 13

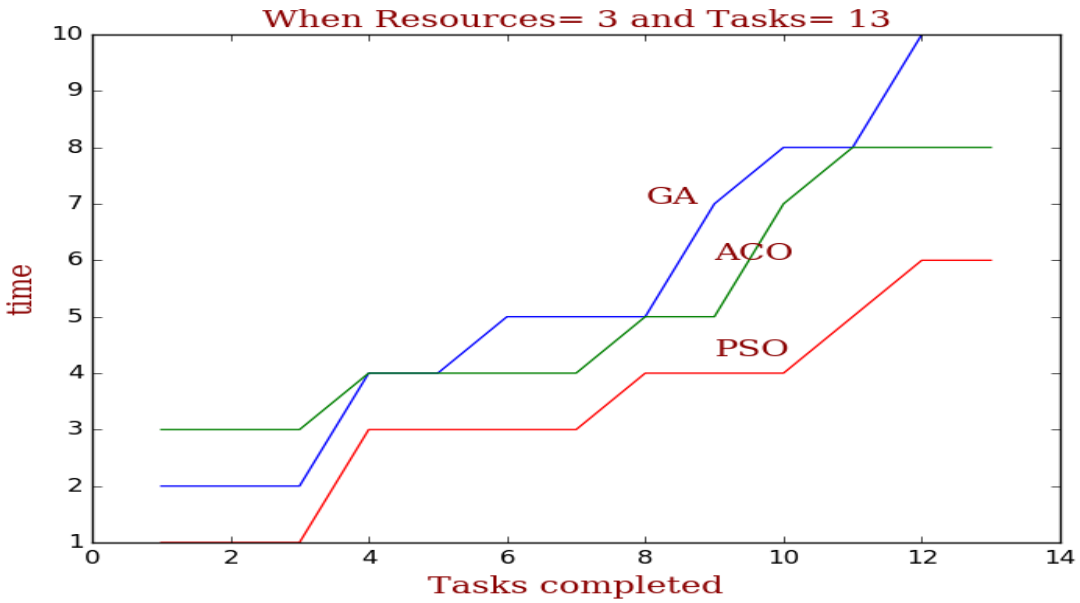

Figure 7: Simulation result 1

Tome	Tasks Completed		
	GA	ACO	PSO
0	0	0	0
1	0	0	3
2	1	0	4
3	3	1	7
4	4	2	10
5	6	8	10
6	9	8	20
7	9	10	25
8	10	11	30
9	12	20	-
10	13	24	-
11	14	25	-
12	15	27	-
13	17	27	-
14	17	30	-
15	24	-	-
16	25	-	-
17	25	-	-
18	30	-	-

Table 2 Resource 5 and Tasks 30

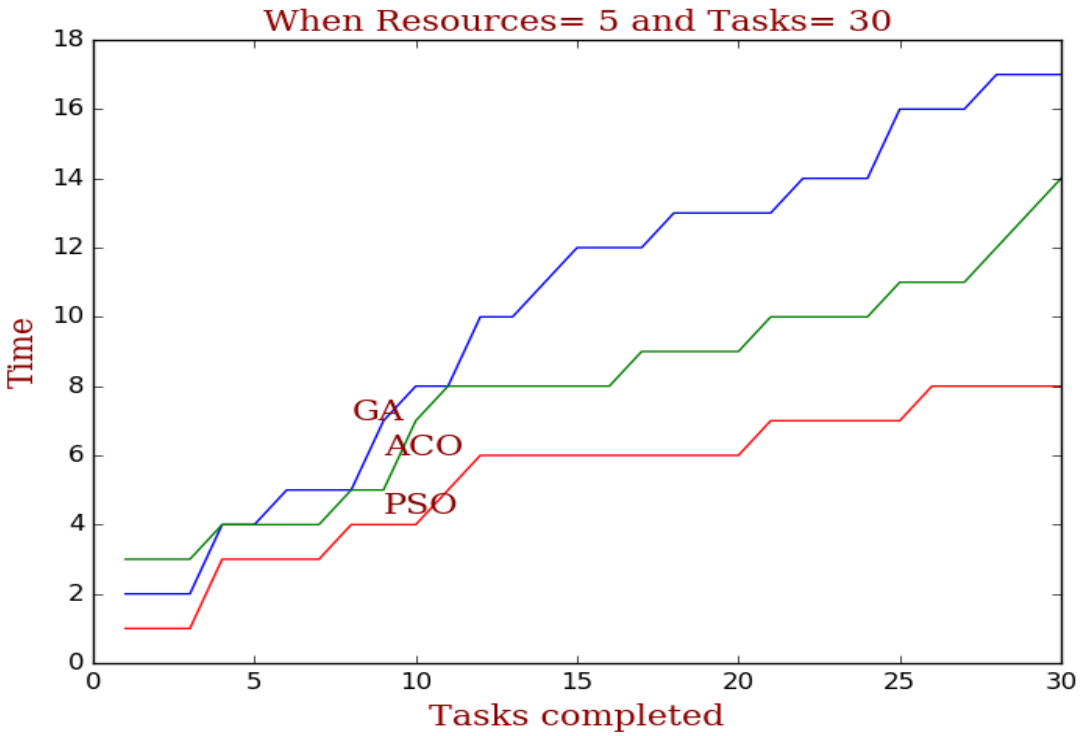

Figure 8: Simulation result 2

Time	Task Completed		
	GA	ACO	PSO
0	0	0	3
1	0	0	3
2	3	0	7
3	3	3	9
4	5	7	9
5	8	9	16
6	8	9	21
7	9	10	26
8	11	16	-
9	11	18	-
10	13	20	-
11	13	23	-
12	17	23	-
13	20	26	-
14	24	-	-
15	24	-	-
16	26	-	-

Table 3 Resource 7 and Tasks 26

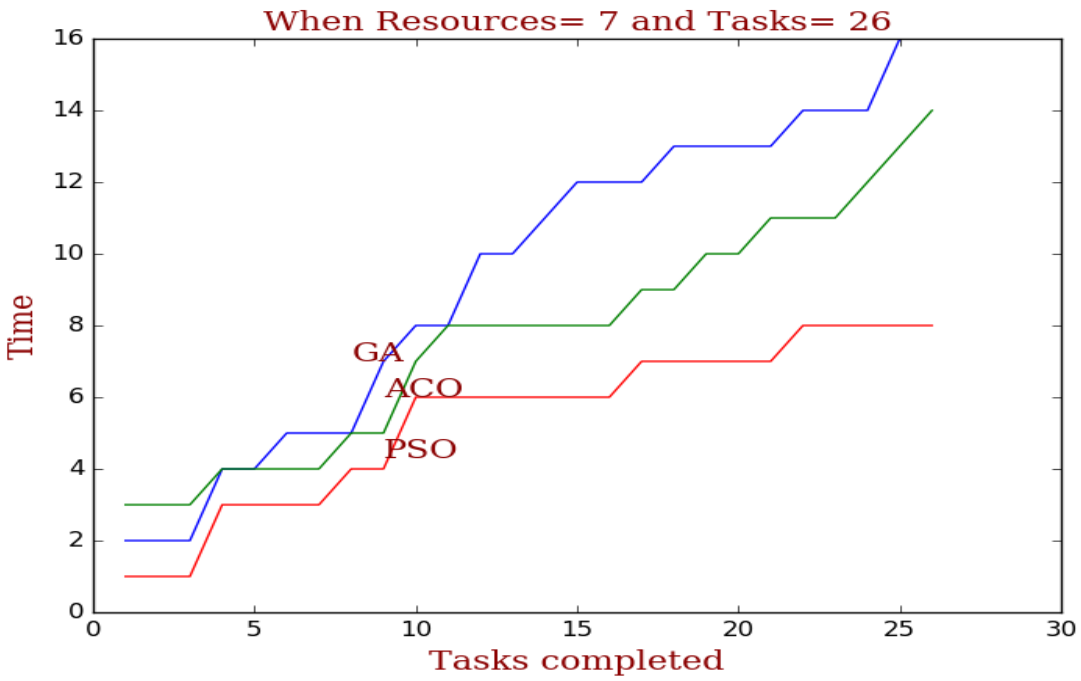

Figure 9: Simulation result 3

4.4.2 Results

From the experiment 1, we found that PSO is 40% better than GA and 25% better than ACO. From the experiment 2, we found that PSO is almost 43% better than ACO and almost 50% better than GA. Similarly, from the experiment 3, we found again that PSO is almost 50% better than GA and 40% better than ACO.

For the huge number of tasks too, we did experiments and almost in every occasion, PSO seemed to be best. It seemed best by an average of 38% better than GA (on an average) and 26.5% better in comparison to ACO (on an average).

CHAPTER 5
CONCLUSION

5.1 Conclusion

Selecting the appropriate resources for the specific task is the one of the challenging work in cloud computing. Here, GA, ACO and PSO are to dynamically generate an optimal schedule so as to complete the tasks in minimum period of time as well as utilizing the resources in an efficient way. The results show that ACO is slightly better than GA in terms of total makespan. In some experiment, the PSO has produced minimum makespan. Evaluation results have shown that PSO seemed to be 38% better than GA and 26.5% better than ACO on average. It is thus concluded that the Particle Swarm Optimization can be efficiently used for task scheduling in cloud. Thus, the Particle Swarm Optimization Algorithm is highly flexible and can be effectively used to schedule task by considering very few control parameters as compared with the other heuristic algorithms.

5.2 Limitation

The further research works that are not addressed in this work and are limitation of this work are also discussed in this chapter. Since the task scheduling problem is NP-complete and its optimal solution is not possible with in polynomial time till now and it is an open question that whether the NP is equal to P or not. So up to now, the NP-complete problems are solved for small instance with sub-optimal solution using some heuristics. Now in this work, the GA, ACO and PSO are used to find the sub-optimal solution.

Since the task scheduling problem is NP-complete, the other heuristic based approach such as modified PSO and BCO can be used to solve the problem with near optimal solution.

5.3 Future Enhancement

The PSO approach can be applied to solve various cloud optimization problems in which there is an uncertainty of task selection. Future developments of this work are related

further improve on algorithm by adding more Quality of Service parameters. In the future, I'm planning to further improve algorithms to re-allocate resources according to requirement or to the existing status of the Cloud in order to optimize resource usage.

References

[1] J. RHOTON, "Cloud Computing Explained. 2.edition," Kent: Recursive Limited, 2011, p. 508 ISBN 9780956355607.

[2] B. C. E. a. L. I. Rimal, "A taxonomy and survey of cloud computing systems," *In Proceedings of the 5th IEEE International joint conference of INC, IMS and IDC,* pp. 44-51, 2009.

[3] G. NEWSROOM, "Gartner Highlights Five Attributes of Cloud Computing," 2009. [Online]. [Accessed 08 05 2011].

[4] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

[5] S. P. G. Reddy, "A Review Work on a Task Scheduling in Cloud Computing using Genetic Algorithm," *International Journal of Science and Technology research,* pp. 241-244, 2013.

[6] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

[7] Q. L. a. L. H. Linan Zhu, "Study on Cloud Computing Resources Scheduling Strategy Based on the Ant Colony Optimization Algorithm," *IJCSI International Journal of Computer Science Issues,* vol. 9, no. 2, pp. 54-58, 2012.

[8] J. K. a. R. C. Eberhart, "Particle swarm optimization," in *in Proceedings of the 4th IEEE International Conference on Neural Networks*, 1995.

[9] S. Poudel, "Comparative Study of Genetic Based Task Scheduling Algorithms in Cloud Computing Environment," Dissertation submitted to Tribhuvan University, Kathmandu, 2013.

[10] A. Pandey, "Comparative Analysis of Genetic algorithm And Ant Colony Optimization Based Task Scheduling," Dissertation submitted to Tribhuvan University, Kathmandu, 2015.

[11] S. P. G. Reddy, "A Review Work on a Task Scheduling in Cloud Computing using Genetic Algorithm," *International Journal of Science and Technology research,* pp. 241-244, 2013.

[12] Y. L. C. L. a. T. C. C.W. Chiang, "Ant Colony Optimization for Task Matching and Scheduling," *IEEE Proc. Computer Digital. Techniques,* vol. 153, no. 6, pp. 373-379, 2006.

[13] Q. L. a. L. H. Linan Zhu, "Study on Cloud Computing Resources Scheduling Strategy Based on the Ant Colony Optimization Algorithm," *IJCSI International Journal of Computer Science Issues,* vol. 9, no. 2, pp. 54-58, 2012.

[14] E. F. e. al., "An Ant Colony Optimization Algorithm for Job Shop Scheduling Problem".

[15] L. S. M. G. a. R. B. Suraj Pandey, "A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environment," *IEEE International Conference on Advance Information Networking and Applications,* pp. 400-407, 2010.